

DECEMBER 1986

COMPUTER

**DATABASE
ARCHITECTURE**



IEEE COMPUTER SOCIETY



THE INSTITUTE OF ELECTRICAL AND
ELECTRONICS ENGINEERS, INC.

REVISED**FTCS**

CALL FOR PAPERS

The Seventeenth International
Symposium on Fault-Tolerant
Computing

JULY 6-8, 1987

SHERATON STATION SQUARE
PITTSBURGH, PENNSYLVANIA, USA

Sponsored by: Computer Society's of the IEEE
Technical Committee on Fault-Tolerant Computing

TC Chairman

T. Anderson
Univ. of Newcastle, England

Symposium Chairman

J. Shen
C M U., Pittsburgh, PA, USA

Symposium Vice-Chairman

D. Siewiorek
C M U., Pittsburgh, PA, USA

Program Co-Chairman

F. Cristian
IBM, San Jose, CA, USA
J. Goldberg
SRI, Menlo Park, CA, USA

Publicity Chairman

B. Bose
Oregon St. Univ., OR, USA

Program Committee Members

J. Abraham, USA
V. Agarwal, Canada
S. Akers, USA
P. Bernstein, USA
J. Gray, USA
R. Gueth, Switzerland
J. Hayes, USA
H. Ihara, Japan
R. Iyer, USA
K. Kinoshita, Japan
J. Knight, USA
H. Kopetz, Austria
L. Lamport, USA
J. Laprie, France
G. Le Lann, France
N. Leveson, USA
B. Liskov, USA
B. Littlewood, UK
E. McCluskey, USA
M. Melliar-Smith, USA
D. Parnas, Canada
D. Rennels, USA
R. Shlichting, USA
F. Schneider, USA
D. Siewiorek, USA
D. Skeen, USA
B. Smith, USA
Y. Tohma, Japan
W. Toy, USA
K. Trivedi, USA

Important Date

December 5, 1986
Submission Deadline

The Fault-Tolerant Computing Symposium has, since 1971, become the most important forum for discussion of the state-of-the-art in fault-tolerant computing. In 1987 the symposium will be held in Pittsburgh, PA, USA. Prof. John Shen, ECE Dept., Carnegie-Mellon University, Pittsburgh, PA 15213, Tel. (412) 268-3601 is the Symposium Chairman.

FTCS17 will address all aspects of specifying, designing, testing, diagnosing and evaluating dependable and fault tolerant computing systems and their components. Particular emphasis will be placed on papers relating to practical experience with real time systems, switching systems and transaction systems as well as the application of fault-tolerance to the design of safety critical systems.

Information for the Authors: Authors should submit 6 copies of papers before the submission deadline December 5, 1986, to the Program Co-Chairman; Dr. Flaviu Cristian, IBM Research K55/801, 650 Harry Road, San Jose, CA 95120-6099, USA, and Mr. Jack Goldberg, SRI Intl, 333 Ravenswood Ave., Menlo Pk., CA 94025. Papers in areas a, b, and f, should be sent to Dr. Cristian, and papers in areas c, d, and e, to Mr. Goldberg.

Papers should be no longer than 5,000 words, should include a clear description of the problem being discussed, comparisons with extant work, and a section on major original contributions. The front page should include a contact author's complete mailing address, telephone number and net address (if available), and should clearly indicate the paper's word count and the area to which the paper is submitted. Submissions arriving late or departing significantly from these guidelines risk rejections without consideration of their merits.

Papers relating to the following areas are invited:

- Design methods and basic algorithms for distributed fault-tolerant systems
- Specification, design, testing, verification of reliable software
- Specification, design, testing, verification, and diagnosis of reliable hardware
- Fault-tolerant hardware systems architectures
- Reliability, availability, safety modeling and measurements
- Fault-tolerant computing systems for safe process control, digital switching, manufacturing automation, and on-line transaction processing



COMPUTER SOCIETY OF THE IEEE



THE INSTITUTE OF ELECTRICAL AND ELECTRONICS ENGINEERS, INC.

COMPUTER

December 1986

Volume 19 Number 12

ARTICLES

7 Guest Editor's Introduction: Future Directions in Database Systems—Architectures for Information Engineering

Nick Roussopoulos

10 Multidatabase Interoperability

Witold Litwin and Abdelaziz Abdellatif

Many users now have an interest in simultaneously accessing several databases. We present the main features of a prototype relational system designed specifically for this purpose.

19 Principles and Techniques in the Design of ADMS ±

Nick Roussopoulos and Hyunchul Kang

A novel architecture based on an incremental access model fully integrates global and local database management. It maintains common access paths on the mainframe and confines uncommon paths to the workstation environment.

26 Metadata Management

Leo Mark and Nick Roussopoulos

A self-describing database system uses an active and integrated data dictionary to provide metadata to systems and users. The data dictionary system uses the services of the database system to manage metadata.

37 Views, Objects, and Databases

Gio Wiederhold

Objects provide the useful abstraction in programming languages; views provide a similar abstraction in databases. Since databases provide for persistent and shared data storage, view concepts will avoid problems occurring when persistent objects are to be shared.

45 Design and Performance Considerations for an Optical Disk-Based Multimedia Object Server

Stavros Christodoulakis and Christos Faloutsos

Optical disk-based multimedia object servers will enable users to search gigabytes of information consisting of attributes, text, images, and voice.

Circulation: *Computer* (ISSN 0018-9162) is published monthly by the IEEE Computer Society: IEEE Headquarters—345 East 47th St., New York, NY 10017; IEEE Computer Society West Coast Office, 10662 Los Vaqueros Circle, Los Alamitos, CA 90720. Annual subscription included in society member dues. Nonmember prices: available upon request. Single copy prices: members \$7.50; nonmembers \$15.00. This journal is also available in microfiche form.

Undelivered copies: Send to 10662 Los Vaqueros Circle, Los Alamitos, CA 90720.

Postmaster: Send address changes to *Computer*, IEEE Service Center, 445 Hoes Lane, Piscataway, NJ 08854. Second class postage is paid at New York, New York, and at additional mailing offices.

Copyright and reprint permissions: Abstracting is permitted with credit to the source. Libraries are permitted to photocopy beyond the limits of US copyright law for private use of patrons: (1) those post-1977 articles that carry a code at the bottom of the first page, provided the per-copy fee indicated in the code is paid through the Copyright Clearance Center, 29 Congress Street, Salem, MA 01970; (2) pre-1978 articles without fee. Instructors are permitted to photocopy isolated articles for noncommercial classroom use without fee. For other copying, reprint, or republication permission, write to Editor, *Computer*, 10662 Los Vaqueros Circle, Los Alamitos, CA 90720. All rights reserved. Copyright © 1986 by The Institute of Electrical and Electronics Engineers, Inc.

Editorial: Unless otherwise stated, bylined articles, as well as products and services offered in New Products and New Applications, reflect the author's or firm's opinion; inclusion does not necessarily constitute endorsement by the IEEE or the Computer Society.



Database Architecture

DEPARTMENTS

4 Letters to the Editor

61 Open Channel:

Let's teach real-world programming

62 Update:

Computer Society election results announced; Taylor Booth dies

64 New Products

66 New Literature

67 Microsystem Announcements

70 Book Reviews:

Microcomputers and Microprocessors; Going from Basic to C

71 Calendar

72 Call for Papers

73 Career Opportunities

82 Annual Index

88 Advertiser/Product Index

88A Reader Service Card

81 Change-of-Address Form



On the cover

The data-collecting function of Mesoamerican architecture is reflected in the ruin of the ancient Mayan observatory Caracol. The structure was built to collect data about such important astronomical events as the arrival of solstices or equinoxes—data vital to the survival of agriculture-based civilizations. Decorations on this and other pre-Columbian architectural marvels often depicted deities and religious rituals, providing archaeologists with important data about the lives of these early people.

Cover photo:
John Curtis—
Taurus Photos

Cover design: Ron DeWilde

In the next issue

Computers for AI applications

COMPUTER

Editor-in-Chief: Michael C. Mulder, University of Portland

Editorial Board

Dharma P. Agnew, North Carolina State University

Jim Haynes, University of California, Santa Cruz

Ronald G. Hoelzeman, University of Pittsburgh

Franklin F. Kuo, SRI International

Francis P. Mathur,

California State Polytechnic University, Pomona

K. M. Mohiuddin, IBM Corporation

Arthur V. Pohl, Iowa State University

Ralph Preiss, IBM Corporation

C. V. Ramamoorthy, University of California, Berkeley

Sellie Sheppard, Texas A&M University

Herry Strong, MTR Corporation

Wing N. Toy, AT&T Bell Laboratories

Herbert Weber, University of Bremen

Heien M. Wood, National Bureau of Standards

Magazine Advisory Committee:

Dennis R. Allison (chair), Dharma P. Agnew, Vishwanath D. Agnew,

James J. Farrell III, Lansing Hatfield, Michael C. Mulder, David Pessel,

True Seaborn, Bruce D. Shriver

Staff

Editor and Publisher: True Seaborn

Assistant Publisher: Gary L. Carter

Managing Editor: Marilyn Potes

Issue Editor: Thomas Szekielwicz

Assistant Editors: Louise Anderson, Nancy Hays

Contributing Editor: Wern Myers

Assistant to the Publisher: Patricia Paulsen

Advertising Director: Michael Koehler

Advertising Coordinator: Carol Porter

Membership/Circulation Manager: Christine Champlin

Art Director: Jay Simpson

Production Supervisor: David Gaines

Design and Production: Miriam Wiegell

Submissions: Submit six copies of all articles and special-issue proposals to Bruce D. Shriver, IBM T.J. Watson Research Center, P.O. Box 218, HO-B04A, Yorktown Heights, NY 10598; (914) 789-7626. For electronic submission, Shriver's addresses are Comppmail + b.shriver@CSnet, Shriver@ibm.com; Vnet, shriver@yktvmh.

IEEE COMPUTER SOCIETY
EXECUTIVE COMMITTEE

President: Roy L. Russo*

Acting-President: J. T. Cain*
University of Pittsburgh
Department of Electrical Engineering
Pittsburgh, PA 15261
(412) 624-5394

Vice Presidents

Publications (acting): Duncan H. Lawrie
Technical Activities (2nd VP): John D. Musa*
Conferences and Tutorials: James H. Aylor
Educational Activities: Glen G. Langdon, Jr.
Membership and Information: Ming T. Liu
Area Activities: H. Troy Naege
Standards: Helen M. Wood

Treasurer: Joseph E. Urban
Secretary: Fletcher J. Buckley
Junior Past President: Martha Sloan*
IEEE Division Directors: Martha Sloan,
Ronald G. Hoelzeman
Executive Director: T. Michael Elliott*
*Ex officio member of Board of Governors

BOARD OF GOVERNORS

TERM ENDING 1986

Dennis R. Allison
Kenneth R. Anderson
P. Bruce Berra
Fletcher J. Buckley
Richard C. Jaeger
Ming T. Liu
Michael C. Mulder
Hillel Ofek
Edward W. Thomas
Joseph E. Urban
Oscar N. Garcia*

TERM ENDING 1987

Barry W. Boehm
Paul L. Borrelli
Glen G. Langdon, Jr.
Duncan H. Lawrie
Susan L. Rosenbaum
Bruce Shriver
Harold S. Stone
Wing N. Toy
Helen M. Wood
Akiko Yamada

PUBLICATIONS BOARD

Dharma P. Agrawal
Vishwanath D. Agrawal
Dennis R. Allison
Bill D. Carroll
Michael Evangelist
James J. Farrell III
Tse-yun Feng
Lansing Hatfield
Ronald G. Hoelzeman
Sam Horvitz

Richard C. Jaeger
Willis K. King
Duncan H. Lawrie
Jack Lipovski
Ming T. Liu
Michael C. Mulder
Theo Pavlidis
David Pessell
C. V. Ramamoorthy
Bruce D. Shriver
Steve Tanimoto

Duncan H. Lawrie Vice President for Publications

SENIOR STAFF

Executive Director: T. Michael Elliott
IEEE Computer Society
1730 Massachusetts Ave., NW
Washington, DC 20036-1903
(202) 371-0101

(Will furnish complete roster
of committees upon request.)

Editor and Publisher: True Seaborn
Chip, Computer Society Press: Chip Stockton
Director, Conferences: William R. Hahngreither
Director, Tutorials: Martez A. Camilleri
Director, Finance: Mary Ellen Gurto

NEXT BOARD OF GOVERNORS MEETING:

Cathedral Hill Hotel
San Francisco, California
February 27, 1987, 8:30 a.m.-5:00 p.m.

THE INSTITUTE OF ELECTRICAL
AND ELECTRONICS ENGINEERS,
INC.

President: Bruno O. Wernschel
President Elect: Henry L. Bachman
Executive Vice President: Emerson W. Pugh
Executive Director: Eric Herz

LETTERS

What is AI?

To the editor:

This is a response to the letter by Myron Kayton in the July issue of *Computer* ("Is the term 'AI' just window dressing?" p. 7).

I agree with him that "it seems that every complicated computer program is being touted as an example of artificial intelligence"; and I also agree that programs written years ago performed expert tasks. However, AI is not "a sales campaign for the Lisp and Prolog languages."

Charniak and McDermott, in their book *Introduction to Artificial Intelligence*, defined artificial intelligence as "the study of mental faculties through the use of computational models." As they pointed out, AI is distinct from psychology. The former is concerned with building working programs that exhibit some kind of intelligence; the program need not use the same method as people. The latter is concerned with understanding and predicting human behavior. Some topics in AI are robotics, vision, natural language, learning, and problem solving. An *expert system* is an application of AI. Expert systems are programs that perform tasks that require expertise but not creativity. An important requirement of an expert system is that it can provide explanation of its line of reasoning, just as an expert can. This is essential for an expert system to gain any confidence from the user. One fundamental issue in AI is *knowledge representation*. In order to build an intelligent system, we must represent knowledge in a form that is unambiguous and can easily be manipulated. AI workers have built various tools to help them do their work. These include languages like Lisp and Prolog and other knowledge representation systems like ART, KEE, Knowledge Craft, and Loops.

Very often, when people say that they have written an AI program, what they mean is that they have written a program using one (or some) of those tools. There are advantages in using these tools to build programs, especially expert systems, because they make it easier to build prototypes and to modify the programs. Whether a program can be called an expert system depends on what it does. If a program provides an expert-

level solution to, or advice on, a complex problem then the program may be called an expert system, irrespective of which language it is written in.

Paul Chung
Artificial Intelligence Applications
Institute
University of Edinburgh

Who could argue against clarity?

To the editor:

I can't quarrel with the desired goal sought by Fletcher Buckley (Letters, Oct. 1986 *Computer*, p.4). Who could argue against clarity?

But I suggest that there are alternate means to that good end. Buckley spelled out my favorite kind of reading: terse, well structured, logical. That style is preferred by analysts, and analysts probably dominate the computer professions. But, though we're likely a majority, there are other styles of thinking that IEEE should not only tolerate, but indeed encourage. Examples include realists, pragmatists, idealists, and synthesists. The well-formed logic of the analyst may constrain some of these other folks. We dare not alienate them; we need their energy, and especially their imaginations.

So, what to do? Rather than seeing a problem here, I see an opportunity. If it's indeed true that an overwhelming majority of *Computer* readers prefer the analytical, logical, orderly style, then perhaps someone should edit and publish a digest written in that style. Articles rewritten in the digest could come from a number of other publications, specifically *not* restricted to IEEE publications; articles could enjoy a variety of styles in their original form, catering to a variety of needs and tastes. IEEE Computer Society members could subscribe to *Computer*, or the digest, or both.

If there is insufficient interest to allow such a digest to succeed, then perhaps we're better off making only minor improvements to *Computer*.

Robert G. Estell
Ridgecrest, California

**CALL
FOR
PAPERS
and Panel
Session Proposals**



**The Eleventh Annual
International Computer
Software & Applications Conference**

compsac87

Tokenawa Prince Hotel, Tokyo, Japan

Tutorials: October 5-6, 1987 Conference: October 7-9, 1987

The conference will bring together computer practitioners, users, and researchers to share their ideas, experiences, and requirements for applications software, management techniques, and software development support.

Papers and panel session proposals related to but not limited to the following areas are invited:

Software Quality and Productivity

- Metrics
- Quality Assurance
- Standards
- Productivity
- Software Configuration Management

Software Engineering Management

- New Management Approaches
- Software Factory Concepts
- Database/Distributed Processing Applications

Development and Maintenance Environments

- Tools and Methodology Support
- Knowledge Based Techniques
- Industrial Environment
- Human Factors

Emerging Technologies

- Parallel Processing
- Distributed Computing Systems
- Supercomputers
- Fault Tolerant Software Systems

Expert/Knowledge Based Systems

- Practical Applications
- Human-machine Interface
- Verification/Testing

**Applications of Software Engineering
Technologies in the areas of:**

- Communications
- Manufacturing
- Banking/Finance
- Medicine
- Airlines

DEADLINES:

- **February 1, 1987** for receiving all papers and panel proposals.
- **March 1, 1987** panel organizers notified of acceptance.
- **March 20, 1987** organizers of accepted panel proposals provide final information on session chairmen and panelists
- **April 20, 1987** authors notified of acceptance
- **June 15, 1987** camera-ready copies of accepted papers and panelists position papers due.

INFORMATION FOR AUTHORS:

- 5 copies of the full paper (double spaced) 1000-5000 words
- Paper should include title, authors/affiliations/address/phone and a 150-word abstract

INFORMATION FOR PANEL ORGANIZERS:

- 5 copies of the panel proposal
- Proposals should include: title, organizer/affiliation/address/phone, names of panelists/affiliations/address/phone, and a 150-word scope statement

SUBMIT PAPERS AND PANEL PROPOSALS TO:

**From Asia, Australia &
New Zealand**

to: Prof. Toshiyasu L. Kunii
Program Co-Chairman
COMPSAC 87
c/o Business Center for
Academic Societies
Japan
Yamazaki Bldg. 4F.
2-40-14, Hongo,
Bunkyo-ku
Tokyo 113, Japan
Phone: 03-817-5831
Fax: 03-817-5836
Telex: 02722268 BCJSP J

From Elsewhere

to: Mr. Albert K. Hawkas
Program Co-Chairman
COMPSAC 87
Sargent & Lundy
Engineering Consultants
55 East Monroe
Chicago, Illinois 60603
U.S.A.
Phone: 312-269-3640
Fax: 312-269-3680
TWX: 910-221-2807
Telex: 280603

Honorary Conference Co-Chairmen

Toshio Sekimoto NEC Japan
Stephen S. Yau Northwestern University USA

Conference Co-Chairmen

Masatoshi Ozeki Information Processing Society of Japan
Nicholas L. Marselos AT&T USA

Program Co-Chairmen

Toshiyasu L. Kunii Tokyo University Japan
Albert K. Hawkas Sargent & Lundy Engineers USA

Program Committee

Moshe Auzan NEC Japan
Joseph Cavetto Rome Air Development Center USA
Dobrosin A. Cernio Rome Air Development Center USA
Ray Chang National University of Singapore Singapore
S. C. Ching GTE Labs Inc. USA
Jung Wei Cho North Atlantic Institute of Science South Korea
Aarti Davis BTG Inc. USA
Lorraine M. David David Computer Technologies Inc. USA
Michael E. Finger IBM Research Center USA
James C. Fennell Gould Research Center USA
Armin Gose Syracuse University USA
Barry Gribben IBM Research Center USA
Robert B. Grafton National Science Foundation USA
Shuichi Harada NTT Japan
Tadato Ichikawa Hitachi Systems Engineering Japan
Rafaela Kerkavani Kyushu University Japan
K.H. Kim University of California Irvine USA

Georgie Kraft De Paul University USA
B. G. T. Lee National Tsing-Hua University Taiwan
Tadato Nakamura Tohoku University Japan
John R. North AT&T USA
Brahmim R. Chryssel Northwestern University USA
Ken Reed Royal Melbourne Institute of Technology Australia
Andrew Rudnicki GTE Communications Systems Corporation USA
Sol M. Shatz University of Illinois Chicago USA
Kouji Tani Osaka University Japan
Wing Yip AT&T Bell Labs USA
Ang-Yeh Tse University of Illinois Chicago USA
Lerry Tubbs U. S. Army Strategic Defense Command USA
Benjamin W. Wahl University of Illinois Urbana USA
Ching Wu Wang Beijing University of Aeronautics Engineering China
Horst Weber Universität Dortmund West Germany
Stan Winkler National Bureau of Standards USA



**THE COMPUTER SOCIETY
OF THE IEEE**



**INFORMATION PROCESSING
SOCIETY OF JAPAN**



**THE INSTITUTE OF ELECTRICAL
AND ELECTRONICS ENGINEERS, INC.**



1987 Winter USENIX Technical Conference Omni Shoreham— Washington, D.C. January 21–23, 1987

TECHNICAL SESSIONS

Wednesday, January 21—
WHAT IT IS TO BE UNIX*

A few simple notions and where they lead. Seven invited talks by leading experts will examine and evaluate the key aspects of UNIX, its evolution, its past and present roles, its various communities, and its successes and failures in meeting their needs and objectives.

Thursday, January 22—
UNIX PERFORMANCE

The day will focus on performance and UNIX systems, including case studies, performance prediction methodologies, techniques for improving system performance, program performance tools and many other related performance issues.

Friday, January 23—
UNIX-BASED DATA MANAGEMENT SYSTEMS

State-of-the-art UNIX DBMS presented by experienced designers and evaluators of applications and databases, as well as the problems associated with data integrity will be explored.

TUTORIALS ON UNIX TOPICS

Presented by leading UNIX experts, the tutorials provide in-depth coverage of essential issues in UNIX technology. The schedule includes: 4.3BSD and System V Internals; Networking Courses on NFS, RFS, and Managing a LAN; Designing Device Drivers; Software Development Using C & UNIX; Adv. Programming In C; Graphics; AI and the UNIX Environment; Performance Measurement/Analysis; Implementing Windowing Systems on Berkeley UNIX; and, Advanced System Administration (System V).

THE SPONSOR

The USENIX Association is a non-profit association of individuals and institutions interested in fostering innovation and sharing ideas, software, and experience where UNIX and UNIX-like systems and the C Programming Language are concerned. USENIX sponsors technical conferences and workshops and an annual vendor exhibition; publishes **login**, a bimonthly newsletter; and serves as coordinator of a software exchange for appropriately licensed members. The individuals and institutional members of USENIX are interested in problem solving with a practical bias, with research that works, and with timely responsiveness within a community made up of executives and managers, programmers and academics.

USENIX

The Professional and Technical UNIX Association

For complete conference information, call: (213) 592-1381 or (213) 592-3243
Or write: USENIX Conference Office, P.O. Box 385, Sunset Beach, CA 90742

The 1987 Uniform Conference, sponsored by /usr/group, will run concurrently with USENIX.
The Uniform Conference will be held at the Washington, D.C. Convention Center.

*UNIX is a registered trademark of AT&T

Guest Editor's Introduction

Future Directions in Database Systems—Architectures for Information Engineering

Nick Roussopoulos, University of Maryland

Has the true "information age" arrived? Earlier this decade, many people were assuring us that its "dawn" had arrived and that it would completely change our lives. The world would be knowledge-driven by accessing accurate information in a snap. The enormous repositories of information and knowledge needed to design and use tomorrow's complicated systems would be found by "letting your fingers do the database walk."

In anticipation of this, there has been a lot more data gathering. Even more than future information systems would be able to digest. Is data gathering sufficient to bring us out of the still gray period of dawn? The answer is no. Data gathering is useful only for "post-mortem analysis." The complexity of today's systems requires engineering information models capable of rapidly recognizing and reusing abstractions acquired from massive amounts of experimental data and from lengthy analyses performed on that data.

How long is dawn going to last anyway? Five years of hardware development got you the processing power of a computer center in a PC for less than \$1000. After more than two decades why can't we make knowledge systems for a reasonable price, even though data and communication, the basic ingredients of knowledge, are as inexpensive as integrated circuits and boards? A 2000-year-old quotation from Aristotle gives a hint: "The whole thing is more than the sum of its parts." It seems that we've got the parts and we can sum them up,

but we fall short of the whole thing. Summation or the gathering of information does not do it.

I dare to predict that the dawn will be definitely gone when we all recognize that the knowledge database is the most fundamental component of any system and that all other components are engineered in and around the management and control of information. It will be gone when the policy makers understand that a lift-off of a shuttle is not the physical movement per se, but the control of all the events preceding it, the controlled decisions that drive it, and the effects of those decisions on future states of it. The physical movement is the result of a set of controlled actions and is as important, or unimportant if you like, as the rotating movement of the wheel of a car during your search for the right exit on a highway in New York City. Only the realization of the importance of information engineering and appropriate funding will bring us into the information age.

In response to the information age vision, database system requirements have changed. The management of complex and continuously evolving factual knowledge is very different from that of the "snapshot" databases captured in commercial systems. The question is: how are we going to evolve from where we are to the database systems of the information age? What are the promising approaches? What are the right directions to pursue? In dealing with trends and predictions, one always runs a risk of missing an important approach. However, the risks of not making such predictions is

greater. It blocks imaginative concepts from being pursued. Ten years ago, when some of us were advocating AI techniques for better information processing, some considered the idea as "illusory" and others as "prophetic." Let's hope that our visions will materialize again.

In this issue

This special issue is focusing on novel database system architectures to support information engineering. Different databases using a variety of forms and special-purpose processing have to be linked. Instead of a cumbersome monolithic architecture that encompasses many independently constructed databases and processors, flexible architectures provide a cooperative environment that is easier and more economical to evolve. Clearly, we cover only a subset of the future directions in databases. We hope, however, that other future issues will deal with the others.

In the first article, Witold Litwin and Abdelaziz Abdellatif discuss the issue of interoperability of multiple databases in a loosely coupled architecture that allows access and manipulation of the databases.

The second article, by Hyunchul Kang and myself, describes a tightly coupled database system architecture that distrib-

utes the database, its accessing, and the application processing between a mainframe and a large number of workstations. It uses a set of bindings between downloaded and mainframe data objects to incrementally maintain the workstation subsets of the database.

In the third article Leo Mark and I present a multilevel architecture that allows data to be modeled and managed by meta-data.

Gio Wiederhold, in the fourth article, discusses an architecture that stores and manipulates engineering information objects.

The fifth article, by Stavros Christodoulakis and Christos Faloutsos, presents an architecture of a system that efficiently stores and accesses multimedia objects.

It is my hope that the articles of this issue will help the reader understand the principles behind the techniques presented. This is important because, if nothing else, these principles have a chance of enduring the evolution of engineering. □

Acknowledgments

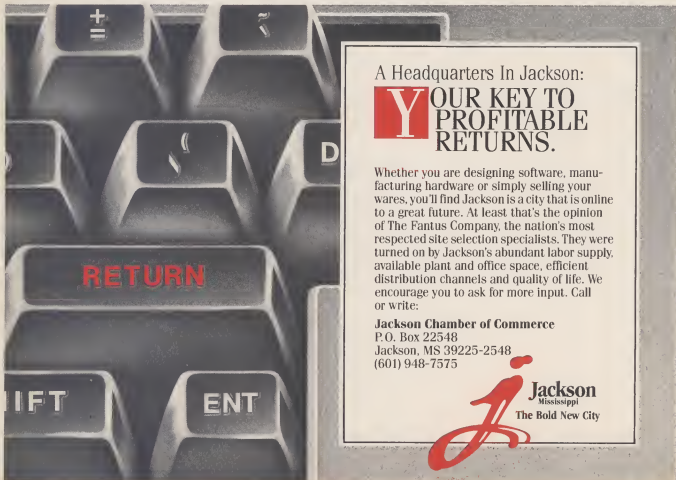
I would like to thank all those who contributed to this special issue. For their support special thanks are due to the editor-in-chief of *Computer*, Michael Mulder, the Editorial Board, and one of its members, Herbert Weber, who was designated to oversee this issue.



Nick Roussopoulos is an associate professor and associate chairman for research and administration of the computer science department at the University of Maryland. In 1985, he was appointed to the Space Science Board Committee on Data Management and Computation, an Academy of Science subcommittee advising NASA. He was the General Chairman of the ACM International Conference on Data Management 1986. He has also organized and chaired a series of workshops for the VHSIC Engineering Information System program. He also serves on the editorial board of the *International Journal of Decision Support Systems*.

Roussopoulos received his BS from University of Athens and his MS and PhD from the University of Toronto. He is a member of the ACM and the Computer Society of the IEEE.

Readers may write to Roussopoulos at the Dept. of Computer Science, University of Maryland, College Park, MD 20742.



A Headquarters In Jackson:

YOUR KEY TO PROFITABLE RETURNS.

Whether you are designing software, manufacturing hardware or simply selling your wares, you'll find Jackson is a city that is online to a great future. At least that's the opinion of The Fantus Company, the nation's most respected site selection specialists. They were turned on by Jackson's abundant labor supply, available plant and office space, efficient distribution channels and quality of life. We encourage you to ask for more input. Call or write:

Jackson Chamber of Commerce
P.O. Box 22548
Jackson, MS 39225-2548
(601) 948-7575

SECOND CALL FOR PAPERS

IEEE INTERNATIONAL CONFERENCE ON COMPUTER DESIGN: VLSI IN COMPUTERS & PROCESSORS

Sponsored by: **IEEE Circuits and Systems Society**
Computer Society of the IEEE
In Cooperation with: **IEEE Electron Devices Society**



ICCD '87

Rye Town Hilton, Rye Brook, New York
October 5-October 8



The International Conference on Computer Design: VLSI in Computers will cover all aspects of the design and implementation of VLSI computer and processor systems. The multi-disciplinary nature of the conference is intended to emphasize the interactions between system design, logic circuit design, memory design, architecture, software, computer-aided design, testing, packaging design and VLSI technology, as they relate to computer and processor system realization.

Original papers are especially solicited in the following areas as they relate to VLSI computer and processor design:

Architecture and Algorithms: Super Computers, Array Processors, VLSI Processors, Computer Networks, Pipelined/Parallel Processors, Concurrent Architectures, Object-Oriented Methods, Optical Computing, Voice Recognition and Synthesis, Data Base Machines, System impact of operating systems and languages.

Computer-Aided Design: Logic/Circuit Simulation, Timing and Layout Analysis, CAD Hardware Accelerators, Graphics and Imaging, Integrated CAD Systems, Distributed Workstations, Formal Verification, Layout and Synthesis, Silicon Compilation, Multiprocessing and Parallel Processing for CAD.

Design and Test: Mixed Design Methodologies, Image/Signal processors, Design for Testability, Design for Self-Test, Testing and Testability Analysis, Fault Modelling, Highly Reliable Computers.

VLSI and Technology: Storage: Optical, Magnetic, Semiconductor, etc.; Wafer Scale Integration, Packaging, Environmental Factors, Chilled CMOS, GaAs.

Computer Applications: Automotive Computers, Robotics, Fifth Generation, Medical Computation, Expert Systems, User Friendly Systems, Computer Security.

Papers are also solicited which describe innovative features of new products, and which describe the overall integration of Technology, CAD, Design and Test and Architecture into the Computer Design Process.

Instructions to Authors

Prospective authors are invited to submit a 1000 word summary. To be considered for review, 1000 word summaries must include the following information: (a) clear description of the contribution and why it is important, (b) for original research submissions state what is novel about the contribution, (c) for review and tutorial submissions state the contribution to the multi-disciplinary mission of the conference.

Summaries are to be submitted in triplicate (3), along with the author names, addresses, affiliations, office and home telephone numbers, by no later than **February 16, 1987 to Technical Program Chair:**

Dr. Prathima Agrawal
AT&T Bell Laboratories
600 Mountain Avenue
Room 3D-480
Murray Hill, N.J. 07974
Telephone: (201) 582-6943

- Proposals for specially organized sessions are solicited. Please forward your proposals to the Technical Program Chairman by no later than **January 14, 1987.**
- Awards will be presented to the best conference papers.
- Notification of acceptance: **April 1, 1987.**
- Final camera ready paper due **July 1, 1987.**



THE COMPUTER SOCIETY OF THE IEEE



**THE INSTITUTE OF ELECTRICAL
AND ELECTRONICS ENGINEERS, INC.**

PLEASE SEND ME MORE INFORMATION ABOUT ICCD '87.

NAME _____

COMPANY _____

ADDRESS _____

CITY/STATE/ZIP _____

COUNTRY _____

SEND TO: ICCD '87

1730 Massachusetts Avenue., N.W.
Washington, D.C. 20036-1903

Multidatabase Interoperability



Witold Litwin and Abdelaziz Abdellatif

Institut National de Recherche en Informatique et en Automatique

Many users now have an interest in simultaneously accessing several databases. We present the main features of a prototype relational system designed specifically for this purpose.

The development of database systems, or DBSs, has given rise to many databases. Frequently, dozens of databases exist on a large computer and thousands of databases are accessible through computer networks. In particular, videotex systems, like Prestel, Teletel, Telidon, etc., provide hundreds of databases on almost any subject such as cinema, train, and airline schedules; banking services; and restaurant fare. An increasing number of users have an interest in simultaneously accessing and manipulating data from several databases. A user may search for restaurants through several restaurant guides or may check several airlines for the cheapest flight, or may need to extract data from a public database for his personal database, etc.

The basic property of such databases is that they are independently created and administered.^{1,3} Since each administrator of a database has his own database needs, databases differ physically and logically. The physical differences may concern data formats, login procedures, concurrency control, etc.⁴ The logical differences may concern data manipulation languages or even entire data models. Even if the participating databases all use the same data model, they usually present mutual semantic conflicts.⁴ These conflicts are differences, redundancies, or incompatibilities with respect to names, values, and meanings among similar data. They result from different perceptions of the same reality by different people. In the sidebar on page 13, we show examples of differences that may occur.

This situation calls for a new type of system designed to manage multiple databases. Such systems have been called

multidatabase (management) systems or MBSS²—a term now rather widespread. One may attempt to base the design of such a system on the idea of global schema. This schema should define from all databases a logically single, integrated database. Users should then manipulate only data of the global schema or of an external schema derived from it. In both cases, they should feel as if they were in front of a classical database for which the global schema would constitute the classical conceptual schema. This is the approach taken, for instance, in MULTIBASE.⁶

However, it appears that the creation of a global schema is usually difficult.^{1,4,7,8} This is the case even if the participating databases constitute only a small number and present the same data model for the common usage. The main reason is the lack of a general solution for the semantic conflicts in a situation in which the autonomy of each of the constituent databases is preserved. In particular, if the databases disagree about a value, then there is no single integrated value satisfactory for all users. Furthermore, no general technique for updates through the global schema seems to exist. Finally, even for organizational reasons alone, a single schema for the thousands of databases on future open systems is a dream.²

A more general approach may be to assume that the databases the user may access basically have no global schema.^{1,2,7,9} The user will then in general know that he faces multiple databases. The system should provide him with functions for manipulating data that may be in visibly distinct schemas and may be mutually nonintegrated. One may say that the con-

stituent databases would then become interoperable, instead of being manipulable only separately or only as if they were components of a single global database.

Interoperability looks appealing for several reasons. At first, it may of course facilitate manipulation of multiple databases when a global schema does not exist. On the other hand, one still may use such a schema when it may be specified (as the schema of a particular view of the entire collection of interoperable databases⁵). Furthermore, because interoperability does not require integration, it should be convenient for administrators, who usually like to remain autonomous (this is often the reason why they choose to create separate databases). Finally, interoperability should also be appreciated by users, who usually like to face a variety of views of reality of, for example, the ratings or opinions given by several independent restaurant guides.

In this article we present a prototype multidatabase system, called Multics Relational Data Store Multidatabase or MRDSM, that is representative of the interoperable approach. The system renders interoperable the databases that are relational or present a relational view for the common usage. The use of the relational model as the common one seems a reasonable choice, as most of the future databases are expected to be of this type or are expected to be provided with the relational interface. The tested databases are those of the well-known MRDS, or Multics Relational Data Store, database system.¹⁰ For MRDS, MRDSM is a user among others. In particular, databases manipulable through MRDSM remain accessible directly through MRDS. Also no change to the MRDS system was introduced. The functions for the translation of other data models into the relational one have not been studied. See for instance Brodie et al.¹¹ for several papers on these issues.

The functions for multidatabase interoperability that MRDSM proposes at present are mainly of two types:

- The administrators of the databases who are cooperating in such a system may define, at the data definition level, names for collections of databases and interdatabase dependencies. The dependencies link the schemas provided for the cooperative usage. These dependencies are outside these schemas, in dependency schemas that define interdatabase relationships with respect to the interdatabase integrity, privacy, or data meanings. Different groups of administrators may define dependencies independently.

- Users have at their disposal the multidatabase manipulation language Multidatabase Data Sublanguage, or MDSL. Unlike the present manipulation lan-

guages, MDSL makes it possible to express retrievals and updates, addressing jointly data in distinct database schemas, as well as to exchange data between databases. The language design is particularly oriented toward the simplicity of a multidatabase query expression. "Simplicity" here means that the user intention becomes a single (formal) query (statement), despite eventual semantic conflicts between the participating schemas. At present, the manipulations the MRDSM user may perform are basically as follows:

- the joining of data in different database schemas;
- the broadcasting of user intentions over a number of database schemas with the same or different naming rules for data with similar meanings;
- the broadcasting of user intentions over a number of databases with data similar in meaning, but with different decomposition into relations;
- the dynamic transforming of actual attribute meanings, units of measure, etc., into user-defined value types;
- allowing data to flow between databases (interdatabase queries); and
- the dynamic aggregating of data from different databases using various new standard (built-in) functions.

Experience with the MRDSM system and everyday use of Teletel databases show that all the above possibilities, at both the data definition and manipulation levels, are highly desirable. Most of them are still unique to MRDSM.

Data definition level

Database schemas. Each database schema presented to MRDSM for multidatabase use is defined through the MRDS data definition language. A schema may be a conceptual schema of an MRDS database, its data model in MRDS terminology, or a database view schema, also called a data submodel. In the latter case, the administrator can hide some parts of the conceptual schema from MRDSM users. The submodel may in particular be secured.¹⁰

Following the terminology of Heimbigner and McLeod,⁴ all such schemas are called export schemas. For the users, export schemas constitute the conceptual schemas of the databases. The users have at their disposal commands enabling them to instantly display the export schemas of databases they wish to acquire knowledge of.

Database access. A database is visible to MRDSM users only after its declaration to the system. The perception the users then have of the database is defined by the export schema. In particular, the access

rights that any user can have to the database are then bound by those defined in the schema. These rights are defined by the administrator through standard MRDS facilities. The administrator retains total control over these rights and, in particular, may change them at any moment. He also may at any moment withdraw the database from MRDSM. The database then becomes invisible again to the MRDSM users.

Multidatabase naming. An administrator or a group of administrators using MRDSM may define for any collection of databases a collective name called a multidatabase name. For instance, the databases Michelin, Kleber, and Gault_M may collectively get the name Rest_guides. Collective names are popular with public database servers. Such names may also simplify the expression of some commands. Otherwise, these commands may require an enumeration of the corresponding databases. A multidatabase name may itself be an element of a larger collection provided with the multidatabase name, etc.

Database identification. Different users may choose the same database or multidatabase name for different databases. The corresponding MRDSM identification rules extend those of MRDS. The MRDSM rules are as follows:

- Any collection of databases managed by MRDSM is called a multidatabase. All databases and, possibly, named multidatabases of the same user implicitly constitute a multidatabase called user multidatabase, the name of which is the user's name. Then, all user multidatabases of the same Multics project implicitly constitute the project multidatabase named as the project is. Furthermore, all project multidatabases in the same MRDSM site constitute a site multidatabase, named INRIA for instance, etc.

- Names given to (multi)databases when they are created are called relative names. The user may refer to his own (multi)databases using only the relative names. To refer to a (multi)database of another user in the same project, the user may prefix the relative name with the corresponding user name. Another possibility is to move into the other user directory, in which case relative names suffice. Similar rules govern access to databases of other projects, etc.

Interdatabase dependencies. The administrators may at present define three types of dependencies:

- manipulation dependencies,
- privacy dependencies, and
- equivalence dependencies.

Manipulation dependencies. A manipulation dependency triggers a query to a database when a given query to another database occurs. For instance, an insertion of data about a new restaurant to one user database may trigger the insertion of the same data to his friend's restaurant database. Manipulation dependencies may thus act as a kind of message passing system. In particular, a triggered query may in turn become the source of another query, etc.

The triggered queries are called complements of the original query. To declare a complement one has to indicate (1) the relation name to be concerned by the source manipulation, (2) the type of the source query (insertion, deletion, etc.), and (3) the name of the Multics segment with the complement itself.

Manipulation dependencies may be defined independently. Through transitivity, a manipulation may then lead to complements outside the original administrator's knowledge. Long and even infinite chains of complements may then appear. MRDSM does not prevent such a situation. However, it limits to a predefined value the number of complements to be processed. The set of complements is determined before the query execution. If the limit, let it be N , is exceeded, a warning is issued. In any case, at most N complements are processed.

While defining the complement, the administrator may indicate whether its execution should precede or should follow the source manipulation. In the former case, the source manipulation is performed only if the complement execution could take place. In the latter case, the administrator may ask for a deferred execution.

Privacy dependencies. The aim of these dependencies is to prohibit manipulations that would match data from different databases in a way that would disclose confidential information. The declaration of such dependencies is similar to that for the manipulation dependencies. However, the privacy dependencies may access the user identification data. They may further correspond to arbitrary checks whose detailed principles are outside the MRDSM design goals. Their basic action is nevertheless to include selections and projections with every query posed by a designated user about a designated relation. These selections and projections have the effect of making certain values invisible.

Equivalence dependencies. These dependencies identify in different databases the primary or candidate keys whose equality of values corresponds to the same real object. For instance, such a dependency could be declared for the

restaurant names and corresponding street names in the case of our example databases. They are useful for formulation of some multidatabase queries with implicit joins. We will present these types of queries in the next section.

MRDSM data manipulation language

An overview. The MRDSM data manipulation language called MDSL extends DSL,¹⁰ the data manipulation language, or DML, of MRDS. Like SQL and QUEL, DSL is based on tuple calculus. New functions within MDSL are mainly intended for multidatabase queries. Some auxiliary functions are designed for query editing, help, and displaying of schemas. Also, any Multics command may be called. Thus, text editors may be called, programs may be compiled or executed, etc. The main functions are designed to make it possible to formulate multidatabase queries of the types discussed earlier. Only these functions will be presented in the following discussion.

Query form. The general form of a MDSL query follows:

```
open name1 [mode1] name2 [mode2]...
-db (abbrev1 name1) (abbrev2
name2)...
<auxiliary clauses>
-range (tuple_variable relation)...
-select <target list>
-where <predicates>
-value value_list
<query commands>
close name1 name2...
```

As usual, the open command opens the databases for processing. The mode argument specifies the opening mode (r for retrieve, u for update in shared mode, and either er or eu [default mode] in exclusive mode). Names may be database names or multidatabase names. The "close" command closes the databases. Both commands are optional if the databases to be used are already open or should remain open for further queries.

The -db clause is also optional. It makes it possible to define abbreviations that may be easier to use. It also makes it possible to define the set of the databases the query should refer to, without closing unused ones (to close a database is usually a heavy manipulation). The databases that a query refers to are called the scope of the query. Open databases constitute the maximal and default scope.

The auxiliary clauses are clauses that do not exist in DSL. They are introduced specifically for the multidatabase environment and will be presented below. The clauses -range, -select, and -where are main clauses. Their syntax is basically

similar to those of DSL. The semantics will be presented below. The -value clause exists only for updates. The value_list is a list of new values. Finally, the query commands are: retrieve, modify, store, delete, copy, move, and replace. The first four commands are DSL compatible. The last three are used for interdatabase queries. Commands may have parameters or clauses, which will be discussed later on.

The names that a query uses for referring to data types are called designators.¹² In DSL and generally in the existing DMLs, designators are unique (unequivocal) identifiers of data types (an attribute, a relation, an entity type, a record type, etc.). If several attributes bear the same name, then the corresponding designators use relation names as prefixes providing unique identifications. If therefore one calls *designator scope* the set S of designated types, then the general rule for relational languages is $\text{card}(S) = 1$ for all S . This is not always the case in MDML, where one may have $\text{card}(S) > 1$. The reasons will be shown later on.

Elementary queries. An MDML query is an elementary query if all designators are unique identifiers of data types within the scope. The result of an elementary query is a relation, or an update to a database relation. DSL queries, as well as queries formulated using known relational DBMSs, are all elementary monodatabase queries. An elementary multidatabase query differs from a DSL query in that designators may concern relations in different schemas. The -where clause of any elementary multidatabase query then involves interdatabase joins. The corresponding implementation issues in MRDSM are presented in Litwin.⁸

Unlike a DSL query, an elementary MDML query makes it possible to use database names as prefixes for unique identification of relations. In the multidatabase environment, it may indeed happen that two relations in different databases bear the same name.

Example 1. Retrieve from My_rest and Cinemas the names of restaurants and of cinemas that are on the same street.

```
open My_rest Cinemas er
-db (m My_rest)(cn Cinemas)
-range(x Rjly cn.C)
-select x.rname y.cname
-where (x.street = y.street)
retrieve
```

The designated relation C is prefixed in order to distinguish it from the relation C within the database My_rest. The mode er is valid for both databases.

Multiple queries

The concept. Multiple queries are introduced for situations where various data-

bases model the same universe. The user may then need to broadcast the same manipulation to several databases (e.g., project any relation describing restaurants on the attribute expressing the restaurant type). Present relational languages do not allow one to simply express such intentions. If only elementary queries are available, then the user needs to formulate as many queries as there are databases. Furthermore, these queries may differ from database to database. In contrast, multiple queries allow one to broadcast the intention through a single query. This may be a considerable simplification, especially for larger scopes. Multiple query has also been called diffusion query or broadcast query.⁷

Formally, a multiple query is a query where some designators designate more than one data type. Basically, these types are in different databases, but they may be in the same database as well. The query is considered as a set of all elementary queries, called subqueries, that may result from all choices of unique identifiers within the designator scopes. A choice may lead to a subquery that cannot be executed. We call the executable subqueries pertinent. The result of a multiple query is basically the set of relations produced by all and only pertinent queries. In MDSL, multiple queries are basically formulated through the application of the new concepts of (1) multiple identifiers, (2) semantic variables, and (3) options on the target list.¹²

Multiple identifiers. A multiple identifier is a name shared by several attributes, relations, or databases. For instance, if the scope of the query is Michelin and Gault_M, then the designator R is the multiple identifier of both R relations and type is the multiple identifier of both type attributes. A multiple query with multiple identifiers is an equivalent of the set of pertinent subqueries resulting from all the combinations of the unique identifiers in the scope of the multiple ones. This function is intended for the broadcast of manipulations of data bearing the same names. Syntactically, the queries are basically formulated as elementary queries. However, the meaning of designators that are multiple identifiers is different.

Example 2. (Q1) Retrieve from Michelin or Gault_M restaurants that are Chinese according to a guide.

```
open Michelin Gault_M
-range (x R)
-select x
-where (x.type = "Chinese")
retrieve
```

This query would be the equivalent of the following two queries:

```
open Michelin
-range (x R)
-select x
-where (x.type = "Chinese")
retrieve

open Gault_M
-range (x R)
-select x
-where (x.type = "Chinese")
retrieve
```

Example databases

The schemas that follow define the databases we use throughout the examples. The relations are defined according to the MRDS data definition language. We avoided the domain and attribute declarations, which are mandatory for actual MRDS schemas. The character * identifies key attributes.

DB Cinemas:

```
C(c#*, cname, street, tel), Cinemas
M(m#*, mname, kind), Movies
P(c#*, m#*, hour, price); Projections
```

DB Michelin:

```
R(r#*, rname, street, type, stars,
  avprice, tel), Restaurants
C(c#*, cname), Courses
M(m#*, c#*, price); Menus
```

DB Kleber:

```
REST(rest#*, name, street, type,
  forks, t#, owner, meanprice),
C(c#*, cname, ncal),
MENU(rest#*, c#*, price);
```

DB Gault_M:

```
R(r#*, rname, street, qual, tel, type,
  avprice),
C(c#*, cname, ncal),
M(m#*, c#*, price);
```

DB My_rest:

```
R(r#*, rname, street, qual, tel, type,
  avprice),
C(c#*, cname, ncal),
M(m#*, c#*, price);
```

The schemas model actual applications, essentially on the French public videotex system Teletel. The database Cinemas models a public database describing the current cinema programs in a city. Michelin, Kleber, and Gault_M model public databases defined upon famous French restaurant guides with the same names (the full name for Gault_M is Gault-Millau). My_rest is a personal database in which a user stores the restaurants of his choice, using as a reference the Gault_M model and data. Some of My_rest restaurants may nevertheless be unknown to Gault_M. Then, some of the restaurants in both databases may be characterized by different values. This would mean that the user replaced in My_rest the Gault_M opinion about a restaurant of his own. Of course he could not do it if he did not have his own database.

The relations within the restaurant databases model respectively restau-

The result would be the set of two relations. Each relation would inherit the database(s) name(s) its attributes come from. The relations would not be union compatible, since their arities and attributes would differ. As guides may disagree upon the type of a restaurant, a restaurant could figure in both or in only one of the relations. The location would then be

rants, courses (dishes), and menus. The attributes are based upon the actual guides. All data are data model homogeneous, as all databases are relational. However, data are to some extent semantically heterogeneous. This is because of the following properties, modeling those of the actual guides and due to the databases' autonomy:

1. Guides partly disagree upon (a) the choice of attributes that should model the universe of restaurants and (b) the names modeling the same concepts.
2. A restaurant may be recommended by more than one guide, but not all restaurants are recommended by all the guides. The situation is similar with respect to courses.
3. Despite the same names, primary key values modeling the same object in different databases are independent.
4. The units and scales of restaurant quality ratings differ from guide to guide. Michelin rates restaurants from none to three stars (*). Kleber ratings are from zero to four forks. Gault_M rating is m/20, where m = 1, ..., 20. There is no objective specific rule for ratings correspondence. Nevertheless, it is frequently clear that guides disagree about a restaurant.
5. The guides may also disagree on the average price of a meal or on a restaurant type. For instance, a restaurant may be Chinese for one guide and Vietnamese for another. The guides may disagree also on the phone number, although it is a candidate key within each database.
6. In particular, Michelin and Gault_M disagree on the meaning of attributes dealing with prices, despite the same attribute names. Michelin prices are without the 15 percent tip, mandatory in France, while Gault_M prices include the tip.
7. In contrast, the guides always agree on a restaurant name and the corresponding street name. This property thus identifies the same restaurant in different guides. Likewise, course (dish) names are the same in different guides.

Similar properties will be typical of the general multidatabase environment. Multiple databases relative to the same universe will usually resemble each other, but will also present numerous semantic differences like those above.

semantically meaningful, as it would implicitly indicate the guide that considers the restaurant Chinese.

Example 3. Delete from Michelin the restaurant with the key $r\# = "456"$.

```
open Michelin eu
-select r#
-where (r# = "456")
delete
```

The query would delete the tuples from all relations in Michelin that have attribute r . It would thus replace two classical relational queries. In addition, it would automatically preserve the referential integrity. This is not the case with queries one may formulate using known relational languages.¹³

Semantic variables. We call a semantic variable a variable whose domain is data type names. In MDSL the domain may be explicit, which means that names are enumerated in an auxiliary clause, or implicit, which means that they result from the variable name. The aim in the concept is to enable the user to broadcast his intention over data named differently. A query may invoke several semantic variables, together with multiple identifiers. Each semantic variable means that the query concerns all the names in its domain. The names may in particular be multiple identifiers. The query is equivalent to the set of pertinent subqueries resulting from possible substitutions of semantic variables and multiple identifiers by unique identifiers.

Explicit domains. The corresponding clause is

```
-range.s (x1.x2. ... xk n1,1.n2,1. ...
          .nk,1 n1,2.n2,2. ... nk,2. ...)
```

Each x is a semantic variable. Each n is a name. The i -th subquery corresponds to the simultaneous substitutions of $n_{j,i}$ to x_j ; $j = 1, \dots, k$.

Example 4. (Q2) Retrieve from Rest_guides, i.e. from Michelin, Kleber, and Gault_M databases, restaurants that a guide considers to be Chinese.

```
open Rest_guides r
-range.s (x R REST)
-range(y x)
-select y
-where (y.type = "Chinese")
retrieve
```

x is a semantic variable whose values are names R and $REST$. Since $REST$ is a unique identifier, the corresponding substitution produces an elementary query. In contrast, since R is a multiple identifier, it leads to two elementary queries, equivalent together to (Q1). All three resulting relations are not union compatible. As for (Q1), they may also contain different restaurants.

Example 5. Retrieve from Cinemas and from My_rest the restaurants and the cinemas that cost less than 30 francs.

```
open Cinemas r My_rest r
-db (cc Cinemas)(mr My_rest)
-range.s (x.y# R.M.r# C.P.c#)
-range (z x) (v y)
-select z
-where (z.y# = v.y#)&(v.price < 30)
retrieve
```

The query will lead to two differently formulated subqueries, one per database in the scope.

Example 6. Change to "123" the phone number "876" in all example databases.

```
open Rest_guides Cinemas My_rest eu
-range.s (t tel t#)
-select t
-where (t = "876")
-value "123"
modify
```

The query will replace five elementary queries. It could in fact replace any number of such queries, provided the database owners agree to name telephone either tel or $t\#$. Thus it is not even necessary for all administrators to agree upon a common name.

Implicit domains. Here, a variable name contains one or more special characters that at present are: *, designating any string of digits, including the empty string, and ?, designating any but only one digit. The domain is then implicitly constituted from all data names in the scope (not data values, like in SQL) that match the resulting pattern. For instance, the domain of $x = R^*$ is all names in the scope that start with R . The subqueries correspond to all pertinent substitutions of data names within the domains. If the characters * and ? are parts of data names themselves, as in R^* for instance, then they should be preceded by the character \. Thus $x = R^*$ would include all names starting with the string R^* .

Example 7. The expression of query (Q2) from Example 4 may be simplified to the following one:

```
open Rest_guides eu
-range(y R*)
-select y
-where (y.type = "Chinese")
retrieve
```

This formulation would remain valid for any number of databases in the scope, provided that all and only restaurant relation names start with the character R . The corresponding query with the explicit domain would then be usually more complex, as it would require a range.s clause with as many identifiers as there are different names.

Options. Current relational DMLs assume implicitly that all the attributes in the -select clause target list are in the schema of the addressed database. As the

examples below will show, it is useful to relax this assumption in the multidatabase environment. The concept of options¹² is intended for this purpose. The corresponding syntax is as follows.

Let d be an attribute designator within the -select clause. Let q be a subquery resulting from some substitutions and a data unique identifier corresponding to d in q .

- If d is preceded by a space, as is usual in DSL, then q is pertinent only if there is attribute a in its scope. Thus, by default a is mandatory.

- d , written $\sim d$, means that q may be pertinent without an attribute named a in the scope. q is then considered as equivalent to a query formulated like q without a in the -select list. The attribute a is thus optional.

- A list $d_1|d_2|...|d_n$ means that the pertinent form of q should contain one and only one a_i . The choice follows the list order. A list preceded with $\sim$ means that the whole list is optional.

Options deal with the existence of attribute names in schemas and not with null values within tuples. However, one may extend this concept to null values as well.

Example 8. Retrieve from Rest_guides name, street, and owner, if any, of all restaurants.

```
open Rest-guides eu
-range(x R*)
-select x.* name x.street ~ x.owner retrieve
```

Since the attribute owner is optional, all three databases will be addressed. If owner were mandatory, the tuples would be retrieved only from the Kleber database.

Example 9. Assume that Gault_M does not have the attribute tel. Retrieve from Rest_guides restaurant names and either phone numbers if available else the corresponding streets.

```
open Rest-guides eu
-range (x R*)
-select x.* name, x!\#.street
retrieve
```

The query will provide the telephone number from Michelin and Kleber, and the address from Gault_M.

Incomplete queries

The concept. While formulating MDSL queries, the user may avoid specifying some equijoins. Basically, one may avoid equijoins linking primary or foreign keys that share a domain. Such queries are called incomplete queries. A subquery of an incomplete query may in particular be an incomplete query. Omitted joins are called implicit joins.¹⁴ They are deduced by the system from database schemas. The result is called a complete query. The completion algorithm is described in detail in Litwin.¹⁴

It is shown that this process leads to the intuitively expected result in more cases than the present algorithms for the universal relation interface.¹³ A major consequence is also that updates may be performed.

This function has as goals (1) to further simplify query formulation and (2) to allow multiple queries to databases modeling the same universe, but different through decompositions into relations. Indeed, there is sometimes no way to express an intention in a single query, if one has to formulate all equijoins corresponding to different decompositions.

Example 10. Retrieve from Michelin the address of all restaurants that serve "confit d'oe".

```
The incomplete query could be
open Michelin er
-select street (Q3)
-where (cname = "confit d'oe")
retrieve

The complete query would be
open Michelin er (Q3')
-range(x R)y M(z C)
-select x.street
-where (z.cname = "confit d'oe") &(x.r#
= y.r#)&(y.c# = z.c#)
retrieve
```

Example 11. Consider now that instead of three relations Gault_M contains only one (universal) relation with all attributes in Gault_M. Assume further that the user wishes to broadcast the query about "confit d'oe" to both Michelin and Gault_M. The formulation (Q3) will then remain valid, provided both databases are open. The clauses will, however, define a multiple query. The query will be the equivalent of two subqueries differing by equijoins. These are (Q3') and the query

```
open Gault_M er
-select street
-where (cname = "confit d'oe")
retrieve
```

Example 12. Delete from My_rest all the courses whose ncal > 2,000.

```
open My_rest eu
-select C
-where (ncal > 2,000)
delete
```

This will delete the appropriate tuples from both C and M.

Dynamic attributes. Dynamic attributes are transforms of actual attributes. They are dynamically defined within a query and unknown to any schema. Except for eventual update limitations, they may be manipulated as the actual attributes.⁹ This function makes it possible for the user to dynamically and subjectively transform data values. Such a need will be frequent in the multidatabase environment. In particular, it will be frequently necessary for interdatabase joins, as joins are meaningful only for data with the same meaning and unit of measure.

The MDSL user may declare a dynamic attribute by means of the following auxiliary clauses:

```
-attr_d [hold] a : C/R
-define by MT(s) = m
-updating s' by MT(s'') = m'
```

a is the dynamic attribute name with value type either C (character) or R (real). If there is no hold argument, then a is known only within the query that defines it. Otherwise, further queries from the user may also refer to a, until the end of the session.

The clause -define by defines the mapping m of actual attribute(s) s on a. It is mandatory for retrievals. MT denotes the mapping type. It may be D for a dynamically defined dictionary (table), F for a formula, or P for a program. The corresponding clause forms are respectively as follows:

```
-define by D(s) = (a1, s1), ..., (ak, sk)
-define by F(s) = arithmetical_formula
-define by P(s) =
Multics_segment_name
```

s_i are actual values and a_i the corresponding dynamic ones. Formulas are arithmetical formulas. The Multics segment contains the program that may be written in any programming language.

The clause -updating defines the mapping of a on an actual attribute s', which is needed when the user updates a. The attribute s' should be one of the actual attribute(s) in s, and s'' are all the other attributes in s, if there are any. This clause is currently mandatory if MT in the -define by clause is P or F. It is optional for D type mappings. The default option is then that a given a value, let it be a', corresponds to the first s_i such that a' = a_i. In all cases, mapping types in both clauses have to be the same.

A dynamic attribute may share the name of an actual one. If some of the actual attributes defining a are not in the scope of the (sub)query, the name in the (sub)query then designates the actual attribute. Otherwise, it designates the dynamic attribute.

The user may also wish to refer to an actual attribute n that shares the name of a dynamic one, previously defined using the hold argument. Then, the -select clause has to be preceded by the clause -actual n.

Example 13. Assume that the "****" rating of Michelin corresponds to Gault_M.qual ≥ 19/20 and "****" corresponds to 17/20 ≤ qual ≤ 18/20. Retrieve from Michelin and from Gault_M restaurants rated "****".

```
open Michelin Gault_M er
-range (t R)
-attr_d stars : C
-define_by P(qual) = star
-select t
-where (t.stars = "****")
retrieve
```

This query leads to two subqueries. The first one to Michelin will select the actual attribute, since the attribute qual, used for the definition of the dynamic attribute star, is not in this subquery scope. The second subquery will produce the values of the dynamic attribute and will use these values for -where clause evaluation with respect to Gault_M. The overall result of the query will be homogenized with respect to the Michelin scale of rating, arbitrarily transposed by the user to the Gault_M database. Note that there is no objective integration rule for Michelin and Gault_M scales or for subjective scales in autonomous databases in general.

star is the program that dynamically computes through the Multics execute command the values of stars. It expresses, in an arbitrary host language, the algorithm:

```
if qual ≥ 19/20: stars = "****" else if
qual ≥ 17/20: stars = "****" endif;
```

The same mapping could also be formulated using the D type declaration as follows:

```
-define_by T(qual) = (****, 20),
(***, 19), (**, 18), (*, 17)
```

Example 14. Retrieve from Michelin restaurants that have the same average price in Michelin and Gault_M.

```
-db (m Michelin) (g Gault_M)
-range (t m.R) (v g.R)
-attr_d price : R
-define_by F(m.r.vprice) = m.r.vprice *
1.15
-select t
-where (t.price = v.vprice) &(t.name =
v.name) &(t.street = v.street)
retrieve
```

The function here renders meaningful the interdatabase join on price, as the meanings of the concept of price differ in both databases. The clauses referring to name and street may be implicit, if the corresponding equivalence dependency was defined. More examples of dynamic attributes as well as the discussion of their implementation in MRDSM are in Litwin and Vigier.⁹

Interdatabase queries. The general form of interdatabase queries is as follows:

```
copy / move
< source selection expression >
store / modify / replace
-target < db_name > . [< relation_name >]
< mapping clauses >
```

The commands copy and move define the action on the source database(s). The copy command copies source data, according to the source selection expression, while the move command also deletes the source data. Its selection expression has then to designate all attributes of a relation. In both cases, if data values

are to be converted, the source selection expression should contain the definition of appropriate dynamic attributes. The meaning of these attributes should be that of the corresponding target attributes. Type conversions, like that of integers to reals, are automatic.

The commands store, modify, and replace define the action on the target. The clause-target identifies the target database or relation. The mapping clauses define the matching of the incoming attributes to the corresponding target attributes. The syntax of the mapping clauses is as follows:

```
<mapping clauses> := -mapping
[<rule>][<matching_list>]
<rule> := by order / by name
<matching_list> := (<option>
[, <option>])
<option> := source_name ->
target_name / target_name /
source_name -> 'new'
```

The source names are the attribute names within the source-select clause. The rule "by name" means that source attributes should be mapped onto target attributes with the same names by default, unless specified otherwise within the matching list. The rule "by order" means in contrast that the attributes should be matched in order of their listing in the source-select clause, to the attributes in the matching list if one is present, or those of the target schema; otherwise, in the prescribed order. In the former case, the elements in the matching list must be target names only.

The matching list alone specifies an arbitrary correspondence. In particular, the option 'new' means that the source attribute does not exist in the target and should be added to the target schema. For security reasons, source attributes without the target counterparts and not declared 'new' are disregarded. Conversely, if a target attribute has no source counterpart, then the corresponding values are set to null or are preserved, depending on the command. Finally, except for the replace command, the query is assumed valid only when the key attributes of the incoming relations correspond to the key attributes of the target relations.

The store command inserts tuples that do not share key values of existing target tuples and preserves those target tuples that share incoming key values. The modify command also inserts incoming tuples without target key counterparts, but it modifies target tuples that share incoming key values. The modification concerns only the attributes that have counterparts within incoming tuples. Neither command affects target tuples whose keys do not share incoming key values. In contrast, the replace command replaces the

whole content of the target with the incoming one.

Example 15. Copy to My_rest restaurants considered as good by Gault_M, as well as the associated courses and menus.

```
copy
-db (g Gault_M) (m My_rest)
-range_s (x g.R g.C g.M)
-range (t x)
-select t
-where qual > 14/20
store
-target m
```

The copy command will produce three subqueries. Two of them will require completion of implicit joins. The whole query will copy three relations, containing respectively the selected restaurants, courses, and menus. The result will automatically preserve the referential integrity. The selected relations and attributes will be mapped on those with the same names within the target. Only tuples that do not share key values already in My_rest will be stored. Thus the user's opinion about a restaurant, a course, or a menu, will have priority. The inverse effect would appear if the modify command were used.

This query represents the case we spoke about in the overview of MRDSM, where source data in several relations should enter several target relations. Some other instances where such a case would arise are a supplier and his parts, a student and his courses, a customer and his accounts, etc.

Example 16. Replace the content of My_rest with the restaurants, the related courses, and menus that correspond to the '***' rating in Michelin. Convert the meaning of the Michelin prices to those with tip included.

```
copy
-db (m Michelin) (my My_rest)
-range_s (x m.R m.C m.M)
-attr_d price : R
-define -f (M.r."price" = m.r."price" * 1.15)
-range (t x)
-select t
-where stars = '***'
replace
-target my
-mapping by name (stars -> qual)
```

Example 17. Consider that the user has changed the schema of the relation C into the following one:

```
C (#*, origin, cal, name),
where the new attribute origin denotes the region or country the course (dish) comes from, if any. The query "copy to My_rest the courses in Gault_M" may be expressed as follows:

```

```
copy
-db (g Gault_M) (m My_rest)
-range (x g.c)
-select x
store
-target m
-mapping in order(c#, name, cal)
```

The values of origin will be null.

Example 18. Consider that the user wishes to keep in My_rest only the best restaurants (those rated more than 16/20). However, he also wishes to save in a separate database, let it be My_rest_archives, the content of relation R. The corresponding query may be as follows:

```
move
-db (m My_rest) (a My_rest_archives)
-select m.R
-where qual < 17/20
store
-target a
```

Standard functions. Standard functions, such as avg, sum, max, etc., may be declared in MDSL in two ways:

(1) Inside the clauses, being then enclosed within square brackets. The function is then evaluated independently for each subquery.

(2) As independent clauses. The function applies then to all the tuples of the query.

Some functions may be applied only to subqueries, some have meaning only as independent clauses, and some may be applied in both manners.

Example 19. Retrieve the average price of meals according to each database in Rest_guides.

```
open Rest_guides er
-select [avg(price)]
retrieve
```

This query returns three values, one for each database. The following returns just one value.

Example 20. Retrieve average price of all meals within Rest_guides databases.

```
open Rest_guides er
-avg
-select price
retrieve
```

In addition to the well known standard functions, the user will need new functions, specific to the multidatabase environment. The following functions should prove particularly useful.

name function. Let *n* be a designator. Then name(*n*) provides the name of data designated by *n*, name(*n*) provides the name of the container of data (relation for an attribute, etc.) designated by *n*, name(.*n*) refers to name(name(*n*)), etc.

This function results from the need for relational operations not only on data values, but also on data names. It may be applied instead of an attribute name within -select and -where clauses. The result is then considered as if it had been an actual attribute value.

Example 21. Retrieve the names of Chinese restaurants and of the guides that recommend them.

```
open Rest_guides er
-range(x R*)
```

```
-select x.*name, x.(name('R'))
-where (x.type = "Chinese")
retrieve
```

Each tuple will then contain the name of a restaurant and the name of the database the tuple comes from. As may be seen, in the multidatabase environment, database names may bear logical information.

norm function. This function merges into one tuple all tuples corresponding to the same real object. The correspondence results from the equality of the keys indicated by the user. Keys may be the primary keys or the candidate keys, if the values of the primary keys within different databases differ. The norm function has the following syntax:

```
-norm (id_att) tuple_var).
```

Here id_att are designators of keys identifying the same object. The function appears only as an independent clause. The result is the natural outer join of the designated relations.

Example 22. Retrieve from Rest_guides Chinese restaurants, creating one tuple per restaurant.

```
open Rest_guides er
-range(x R")
-norm("name, street) x)
-select x
-where (x.type = "Chinese")
retrieve
```

The result would be a single relation R whose attributes would be all those of involved relations, prefixed with the database name in the case of a name conflict. The id_att would figure only once, according to the definition of natural joins. To any restaurant would correspond exactly one tuple. The values of the attributes corresponding to databases that do not recommend the restaurant would be null. Absence of null values in at least some columns corresponding to attributes from a database would thus be an implicit indication that the corresponding guide recommends the restaurant.

Outer joins are unknown to MRDS. Algorithms for efficient processing of such operations have therefore been investigated. Discussion of such algorithms may be found in Dayal.¹⁵

upto function. This function appears only as an independent clause. It limits the multiplicity of information that may come from several databases. For instance, a query to 10 restaurant databases may ask for at most two recommendations of a restaurant. In particular the user may give priority to databases he trusts more than others. The function syntax is as follows:

```
-upto (n (A) [B]).
```

The function provides at most $n \geq 1$ tuples sharing the values of attributes designated in the list A. Priorities corre-

DISTRIBUTED SYSTEMS SPECIALISTS

Tomorrow's Computing
Technology is Today's Challenge

at
IDA

Some of the nation's most exciting developments in software technology, supercomputer architecture, AI, and expert systems are under scrutiny right now at the Institute for Defense Analyses. IDA is a Federally Funded Research and Development Center serving the Office of Secretary of Defense, the Joint Chiefs of Staff, Defense Agencies, and other Federal sponsors.

IDA's Computer and Software Engineering Division (CSED) is seeking professional staff members with an in-depth theoretical and practical background in the area of Distributed Systems. Tasks include efforts aimed at designing and prototyping systems, evaluating/assessing alternative strategies or designs, and advising major DoD programs on the suitability of various technologies or on the need for research in the field.

Specific desired interests and skills include:

- Operating systems — including clock synchronization, efficient IPC and RPC, centralized and distributed scheduling, and resource allocation
- Data base systems — including access control, distribution transparency, data redundancy, concurrency control, security
- Communication protocols
- Reliability and survivability — including issues related to

transaction logging, rollback and recovery, process migration, and system reconfiguration

Specialists in other areas of Computer Science are also sought: Artificial Intelligence and Expert Systems, Software Engineers, Computer Security Scientists and Programming Language Experts.

We offer career opportunities at many levels of experience. You may be a highly experienced individual able to lead IDA projects and programs . . . or a recent MS/PhD graduate. You can expect a competitive salary, excellent benefits, and a superior professional environment. Equally important, you can expect a role on the leading edge of the state of the art in computing. If this kind of future appeals to you, we urge you to investigate a career with IDA. Please forward your resume to:

Mr. Thomas J. Shirhall
Manager of Professional Staffing
Institute for Defense Analyses
1801 N. Beauregard Street
Alexandria, VA 22311

An equal opportunity employer.
U.S. Citizenship is required.



spond to the order of the list B that designates database names. A, n, and B are optional. If A is not specified, the query processing stops after a non-null response from n databases. The default value of n is 1. Finally, empty B means that the user has no preference.

Example 23. Retrieve the number of calories of "confit d'oie", preferably according to Kleber.

```
open Rest_guides er
  upto (1 (cname) [Kleber])
  -select ncal
  -where (cname = "confit d'oie")
  retrieve
```

The result will come from another database only if Kleber does not describe confit d'oie.

Example 24. Retrieve all Chinese restaurants from Rest_guides, but limit the number of descriptions of each restaurant to two. Give the priority to Michelin and Kleber descriptions.

```
open Rest_guides er
  -range(t R*)
  upto (2 (t.name t.street) [Michelin Kleber])
  -select t
  -where(t.type = "Chinese")
  retrieve
```

A restaurant will figure in the output from Gault_M only if it was not in the output from Michelin or Kleber.

As databases become easily accessible on computers and networks, more and more users face multi-databases. New functions for data manipulation languages are then needed, as present languages were designed for manipulations of single databases. These functions should especially allow users to manipulate autonomous databases. MRDSM offers some such functions for relational databases. Most future distributed databases will be of this type or will at least present a relational view. Numerous examples show that the MRDSM functions should prove useful for a large variety of user needs. This is because of their flexibility and open nature with respect to the accommodation of autonomous names, values, and structures. Most of these functions are not yet available in other languages and systems. The functions designed for MRDS databases can also be added to other relational languages. The corresponding extension of SQL is under study. The concepts of multiple identifiers, semantic variables, inter-database queries, etc., may further be applied to other data models. They may thus form a useful basis for the design of distributed systems using other popular models.

In particular, several functions have also proved useful in a single database

context. Thus, the concept of the multiple identifier may help in preserving the referential integrity. Implicit joins simplify the formulation of most relational queries. Dynamic attributes are useful for applications where subjective or frequently changing value mapping rules render the traditional concept of view too static. It should thus be worthwhile to incorporate similar functions in any relational system.

The functions discussed lead to many open problems at the implementation level. Also, new functions may be added. A function currently studied for implementation concerns multidatabase views, to be defined through stored MDSL queries, in the manner similar to those of DB2 and of INGRES. Knowledge processing techniques are also investigated, as they should prove particularly useful in the multidatabase environment.^{2,9} In particular, they should enlarge the class of intentions expressible as a single query and should make it possible to further simplify the expression of some queries. □



Witold Litwin has been research director at the Institut National de Recherche en Informatique et Automatique, or INRIA, since 1977. He is mainly interested in the design of multidatabase management and distributed systems and in dynamic hashing data structures. He currently leads the SESAME project at INRIA. He is also the author of several papers, the editor of three books, and a referee for ACM, IEEE, and NSF.

Litwin received his PhD in Computer Science in 1971 and Ès-Science doctor degree in mathematics from the University of Paris 6 in 1979.

Readers may write to Litwin at INRIA, BP 106, 78153 Le Chesnay Cedex, France.



Abdelaziz Abdellatif is a doctoral candidate in computer science at the University of Pierre et Marie Curie, Paris. Since 1983 he has been participating in the design of the multidatabase system MRDSM at SESAME Project at the Institut National de Recherche en Informatique et Automatique.

Abdellatif received the Diplôme Universitaire des Études Scientifiques in 1979 and the engineering degree with a computer science specialization in 1983 from the University of Tunis. In 1984, he received the Diplôme des Études Approfondies in computer systems from the University of Pierre et Marie Curie.

Readers may write to Abdellatif at SESAME Project, INRIA, BP 105, 78153 Le Chesnay Cedex, France.

References

1. D. Helmigier and D. McLeod, "A Federated Architecture for Information Management," *ACM-TOIS*, Vol. 3, N3, 1985, pp. 253-278.
2. C. Hewitt and P. De Jong, "Open Systems," *On Conceptual Modeling: Perspectives from Artificial Intelligence, Databases, and Programming Languages*, eds. M. L. Brodie, J. Mylopoulos, and J. W. Schmidt, Springer-Verlag, Berlin, 1985, pp. 147-164.
3. P. Selinger et al., *The Impact of Site Autonomy on R*, Databases-Role and Structure*, Cambridge Univ. Press, New York, 1984, pp. 151-176.
4. S. Ceri and G. Pelagatti, *Distributed Databases Principles & Systems*, McGraw-Hill, New York, 1984.
5. W. Litwin et al., "SIRIUS Systems for Distributed Data Management," *Distributed Databases*, ed. H. Schneider, North-Holland, New York, 1982, pp. 311-366.
6. T. Landers and R. L. Rosenberg, "An Overview of MULTIBASE," *Second Symp. Distr. Databases*, Berlin, Sept. 1982, North-Holland, New York, 1982, pp. 153-184.
7. P. Lyngbaek and D. McLeod, "An Approach to Object Sharing in Distributed Database Systems," *Vldb 83*, Florence, Oct. 1983, pp. 364-376.
8. W. Litwin, "An Overview of the Multidatabase System MRDSM," *ACM Nat'l Conf.*, Denver, Oct. 1985.
9. W. Litwin and Ph. Vigier, "Dynamic Attributes in the Multidatabase System MRDSM," *IEEE-CompDec-2*, Los Angeles, Feb. 1986.
10. *Multis Relational Data Store (MRDS) Reference Manual*, CII-Honeywell bull. ref. 68 A2A55 REV4, Jan. 1982.
11. *On Conceptual Modeling: Perspectives from Artificial Intelligence, Databases, and Program-*

- ming Languages*, eds. M. L. Brodie, J. Mylopoulos, and J. W. Schmidt, Springer-Verlag, New York, 1984.
12. W. Litwin, "Concepts for Multidatabase Manipulation Languages," *JCIT-4*, Jerusalem, June 1984, pp. 433 and 442.
 13. J. D. Ullman, *Principles of Database Systems*, second ed., Computer Science Press, Rockville, Md., 1983.
 14. W. Litwin, "Implicit Joins in the Multidatabase System MRDSM," *IEEE-COMPASAC*, Chicago, Oct. 1985.
 15. U. Dayal, "Query Processing in a Multidatabase System," *Query Processing: Database Systems*, ed. Kime et al., Springer-Verlag, New York, 1985, pp. 81-108.

Principles and Techniques in the Design of ADMS $\pm$



Nick Roussopoulos and Hyunchul Kang
University of Maryland

A novel architecture based on an incremental access model fully integrates global and local database management. It maintains common access paths on the mainframe and confines uncommon paths to the workstation environment.

Local area networks and workstations are now driving the hardware market. Homogeneous software—operating systems with remote file access and procedure calls, software interconnection buses, and other integrated running environments—are defining new system architectures distributed over heterogeneous hardware. Emerging from this trend is the powerful concept of cooperative coupling between a mainframe and a workstation. This coupling will generate a wide class of software systems in which the mainframe provides high-performance resources and off-site synchronization, while the workstation provides privacy, independence, and tailoring of the applications.

ADMS $\pm$ ¹ is one such advanced database management system. Its architecture harmoniously integrates a mainframe database system, called ADMS+, and a large number of workstation database systems, called ADMS-. ADMS+ is a full-fledged system that, in addition to its processing tasks, keeps track of the subsets of the database downloaded to the workstations. ADMS- is a trimmed version of the same database system that has neither a security nor a concurrency control subsystem because it operates in a single-user mode. ADMS- can be thought of as an intelligent cache database subsystem that capitalizes on the locality of the downloaded data and the ability to process

it independently from ADMS+. The communication in ADMS $\pm$ is between the workstations and the mainframe. No communication exists between workstations, which are typically turned off when not in use.

Design goal and major advantages

The principal design goal of the ADMS $\pm$ architecture is to download to each workstation a tailored subset of the database to be processed locally by special-purpose transactions running on the workstation. The mainframe maintains the complete database and additional access aids for subsets shared by several workstations. As the workstation user accesses the databases through ad hoc or precompiled queries, a local subset of the database is dynamically built and incrementally maintained on his workstation. The local database provides very quick access because transactions run independently from the mainframe. And since a lot of the query processing is done locally, the mainframe load is reduced considerably.

ADMS $\pm$ is founded on ADMS,² the first system that uses an *incremental access model*. Contrary to traditional reexecution of access paths,³ ADMS incrementally maintains and uses previously found

access paths without repeating the search. Updates are logged but not broadcast to the affected access paths. Instead, when a possibly outdated path is to be used for answering an ad hoc query, it is incrementally updated and cached. This delayed update strategy incurs no overhead for path maintenance. And, unless the backlog of increments is extremely long, the cost of answering a query based on a possibly outdated access path—that is, the cost of updating and simultaneously caching the path—is smaller than the cost of reexecuting the query.⁴ Therefore, all access-path maintenance cost is subsumed in the cost of answering queries based on those paths. Furthermore, this cost is only paid once. Subsequent queries sharing the same access paths do not have to process already processed increments.

The basic philosophy of ADMS is to integrate and efficiently support shared access paths by *deprecating* the incremental update cost against future use of the paths. ADMS $\pm$ extends this philosophy by *globalizing* paths common to a large number of workstations and by *localizing* less-common paths on the requesting workstations. The global paths are the ones worth deprecating. Uncommon and very specialized paths are confined to their own workstation environment and thus do not affect the rest of the system. *Promotion* of paths from local to global and *demotion* from global to local is done dynamically in ADMS $\pm$. The parameter controlling globalization/localization can be configured differently depending on the database requirements.

A basic assumption of this architecture is that, unlike communication network lines, the lines between the mainframe and each workstation have high bandwidth with no delays, similar to those connecting terminals to a mainframe. The cost of downloading data to the workstation is the same as that of dumping the data to the terminal. The only additional cost comes from storing the downloaded result on the workstation, but this one-time cost does not affect the mainframe. This, and the fact that no overhead is incurred for local path maintenance, permits attachment of numerous workstations to ADMS $\pm$. In theory, since much of the processing is downloaded, a lot more workstations than terminals can be supported for a given mainframe work load.

The main advantages of the ADMS $\pm$ architecture are outlined below:

(1) The response time of locally processed queries is dramatically decreased because the workstation runs in a single-user mode and no dynamic security checking is necessary for the downloaded portion of the database. Security is checked only once, on the mainframe before downloading.

(2) Although ADMS $\pm$ provides an extended centralized database environment, it distributes data and processing to the workstations in a simple and powerful manner that avoids the difficult problems of concurrency and data consistency control of fully distributed environments.^{5,6}

(3) Data is distributed dynamically according to workstation requests. The user interacts with ADMS $\pm$ as if he were using a centralized system. Initially, no data is stored on the workstations; it is downloaded on demand and incrementally maintained as the system processes workstation requests.

(4) The deferred update strategy with no broadcasting drastically reduces the overhead from message traffic to synchronize the updates. In an ordinary distributed DBMS architecture, the message traffic to obtain the appropriate locks is four times the number of sites.⁷ Therefore, even for a moderate number of workstations, the message traffic is very high.

(5) ADMS $\pm$ architecture is very modular and extensible. Very little preparation is necessary before adding more workstations. Furthermore, the additions incur very little overhead because most of the incremental update cost for local database subsets is paid by the workstations.

(6) The user can speed query response by accessing "almost up-to-date" data instead of "up-to-moment" data. In this case, his queries are processed locally on the workstation without having to wait for the current updates.

Global and local access path separation and binding

ADMS $\pm$ supports the access path model defined in Roussopoulos.^{8,9} According to that model, an *access path* is defined by a *query graph* whose nodes are base relations and/or views, and whose edges are relational operators, such as select, join, union, and intersection. Each node in an access path is either a preexisting base relation (or view) referenced by the query or an intermediate result needed to generate the target relation. The lowest node on the query graph is the *target relation*. We refer to the base relations and/or views on an access path by *data objects*.

In ADMS $\pm$, the access path defined by a query on a workstation may refer to mainframe and/or workstation base relations and/or views. Thus, this access path is separated into a *global subpath* that accesses data objects on the mainframe and the *local subpath* that accesses data objects on the workstation. The target relation of any path is, by definition, a local subpath of length zero. Therefore, there is

always a local subpath to every access path and, in some cases, the whole access path is local.

The connection of the local subpath to the global subpath is defined by means of a set of *bindings* between mainframe binding data objects and workstation bound ones. The extension required by ADMS $\pm$ to the access path model of ADMS deals with these bindings.

There are two types of bindings between mainframe and workstation objects. The first one is formed by downloading a base relation or a view from the mainframe to the workstation. We call this *identity binding*. In Figure 1a, R is a global base relation and $M(R)$ represents a downloaded copy of R ; the horizontal line separates the mainframe and the workstation. A mainframe view V , on the other hand, which consists of pointers, is materialized first and its materialization $M(V)$ is downloaded. $M(V)$ has identical storage structure with a base relation. This is necessary to make access to $M(V)$ local in the workstation.

The second type of binding results from deriving a view to be stored on a workstation. We call this form of binding *derivation binding*. This is typically the result of selections and/or joins from mainframe base relations and/or views. The derived workstation view is also materialized so that access to it is done locally on the workstation. Figure 1b shows the unary and the binary derivations of views from the mainframe to a workstation. For the binary derivation, either both operand relations are on the mainframe or one of them is on the mainframe and one on the workstation. The derived view $M(V)$ is bound to its operand relations on the mainframe.

New access paths derived from other mainframe data objects are downloaded to a workstation and become local. Local paths are uploaded when they are found to be shared by several workstations. While downloading creates new bindings, uploading modifies existing bindings. The operation that breaks existing bindings is the removal of a workstation object bound to one or more mainframe objects.

The bindings define the points of global and local subpath separation, that is, they are points of transfer of control and processing from global to local. When an access path defined by a query on the workstation is to be executed, the global subpath is processed first, followed by the update of the path's bindings, followed by the processing of the local subpath. The path's bindings are updated by reflecting the changes made on the mainframe's binding objects to the workstation's bound objects. When this is complete, control is transferred to the local subpath process.

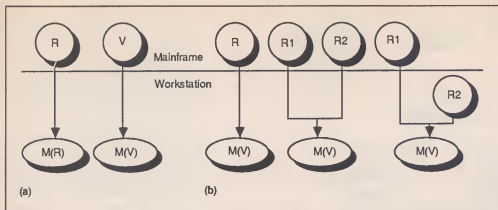


Figure 1. Bindings of data objects between mainframe and workstation: (a) identity binding; (b) derivation binding.

Incremental updates of downloaded objects

In ADMS+, identity and derivation bindings are updated incrementally. The advantage of incremental updating over downloading a fresh copy of the bound object is that, when only a few changes have been made on the mainframe's object, the existing object on the workstation is almost correct to cache and only minor adjustment is necessary. Incremental updating is cost-effective when the number of changes is smaller than a given threshold beyond which the cost of incremental update is more than the cost of downloading a fresh copy. Threshold values will vary with specific implementations and communication parameters.

Incremental updating requires that changes to each base relation and/or view on the mainframe be recorded on a backlog, which can then be transmitted to a requesting workstation. A backlog of a relation or a view consists of a list of triplets of the form $\langle \text{operation}, \text{tuple-id}, \text{DI-log} \rangle$ where "operation" denotes whether the tuple-id was inserted or deleted. A modification is handled as an ordered pair of a deletion followed by an insertion. DI-log is either a pointer to a file D-log, which stores the deleted tuples, or a pointer to a newly inserted tuple. The deferred update policy necessitates storing the deleted tuples for updating the affected access paths and secondary indexes.

The backlogs are the basis for the incremental updates. All the increments of the backlogs between two consecutive requests are used. These are the *backlog differential files*. Since different workstations request backlog differentials at different times, they are responsible for providing, along with the request, a pointer to the last backlog entry downloaded. This frees the mainframe from keeping track of the state of the numerous workstation databases.

Since backlogs in the mainframe grow continuously, unnecessary entries must be

removed periodically. A portion of the backlog can be erased, provided all workstations have requested differentials beyond that point. Keeping a counter of the number of workstations for each insertion or deletion would, however, entail a great deal of overhead. A more practical approach is to time stamp the backlogs at discrete points. For each backlog and time stamp, a counter tracks the number of workstations requesting differentials past the time stamp. When the counter reaches the total number of workstations attached to ADMS+, the portion of the backlog prior to the time stamp can be eliminated.

Another practical approach is to clean up backlogs off-line. This can be combined with a forced update propagation of all backlog differentials to a workstation when its user is off the database. This will guarantee an up-to-date version of the downloaded database at login time and smaller backlogs on the mainframes. Backlog cleaning techniques and policies are also application and implementation dependent.

Incremental update of the identity binding. The two identity bindings—the downloaded base relation and the downloaded view—are updated by different methods because of differences in their storage structures.

Downloaded base relation. A base relation's storage structure is identical whether on the mainframe or on a workstation. Let R be a mainframe relation and $M(R)$ be the identity function that materializes R on a workstation. Two slightly different methods can be used to incrementally update R from the backlog differential $d(R)$ of R . The first is by transmitting to the workstation those pages of R that are found dirty, that is, when there exists at least one entry in $d(R)$ pointing to them. The other technique is by transmitting $d(R)$ itself and applying all its recorded changes to $M(R)$.

The trade-off involved is between (1) the time to retrieve, transmit, and restore

the dirty pages and (2) the time to transmit the backlog differential and the workstation CPU time to repeat the updates. The local write cost of the dirty pages is the same for both techniques.

Downloaded view. A view on the mainframe ADMS+ uses a pointer array for a storage structure. This structure is very efficient. It occupies less space than its corresponding materialization, and it can be directly cached without search. A downloaded view, however, cannot point to external records and must be stored in its materialized form and accessed locally.

Let V be a view on the mainframe and $M(V)$ its materialized version on a workstation. Let also $d(V)$ be the differential file of V on the mainframe. To incrementally update $M(V)$, $d(V)$ has to be transmitted in its materialized form, and all changes in $d(V)$ have to be reflected on $M(V)$. $M(V)$ has to be searched with the values found on $d(V)$ for tuples that must be deleted. Insertions require no search. They are just appended to $M(V)$.

Incremental update of the derivation binding. We discuss here the two most important derivation bindings, the selector and the join. Similar incremental algorithms can be easily derived for other relational algebra operations such as union and intersection.

Selector. Let $M(V)$ be a materialized view on a workstation derived by a selection operation on a base relation R (or view) on the mainframe using a qualification Q . For the sake of presentation, we first describe the incremental update of the view V as if it were stored in the mainframe. Deletions found in $d(R)$ are deleted from V by comparing tuple-ids without looking at the values of the deleted tuples. Insertions are checked against Q , and if they qualify, their tuple-ids are inserted in V .

For a materialized view $M(V)$ on a workstation, the update is done a little differently. Instead of $d(R)$, a materialized

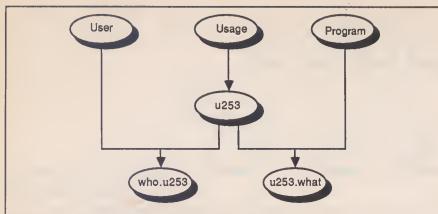


Figure 2. Two access paths with a common subpath.

copy of $d(R)$ must be transmitted to the workstation. Actually, of all deletions and insertions in $d(R)$, only the ones satisfying the qualification Q need be transmitted. Then, a search on $M(V)$ finds and removes all deletions from $M(V)$. Qualifying insertions are inserted into $M(V)$ without search.

Join. Again here, we present the principle of the incremental update of a view on the mainframe and then extend it for the downloaded joins.

Let V be the join of two relations $R1$ and $R2$ (the algorithm is the same for views). First, the deletions found in $d(R)$ and/or $d(R2)$ are deleted from V by comparing tuple-ids again without looking at the values of the tuples. For the insertions, those inserted in $R1$ must be joined with $R2$. We call this *small join* because it only joins the newly inserted tuples of $R1$ with $R2$. Similarly, $R1$ is small-joined with the insertions of $R2$. The addresses of the tuples generated by the two small joins are inserted in V . Some bookkeeping is necessary to avoid the duplication of joining tuples constructed from tuples inserted in both relations.

Let $M(V)$ be a materialized view on a workstation derived from a join of two base relations $R1$ and $R2$ residing either both on the mainframe or one on the workstation and the other on the workstation. We first describe the incremental update for the most general subcase when both reside on the mainframe.

The algorithm is very similar to the incremental algorithm for V , except that the deletions can only be done on $M(V)$ by transmitting materialized tuples instead of addresses. For the insertions, the same small joins are done on the mainframe, and the generated tuples are downloaded and inserted in $M(V)$.

If one of the two operands, say $R2$, is on the workstation, the deletions of $R1$ must

be sent to the workstation in their materialized form for use with $d(R2)$ to delete corresponding tuples of $M(V)$. Since the materialized deletions contain values but no addresses, the tuples are found by search. No search based on tuple values is necessary to take care of the deletions of $d(R2)$.

Insertions require the same two small joins. The insertions of $R2$ are transmitted to the mainframe to be small-joined with $R1$. The other small join is done on the workstation, where transmitted insertions of $R1$ are small-joined with $R2$. All generated tuples are inserted in $M(V)$.

Deferred update strategy and concurrency control

Deferred update. The ADMS family of systems has adopted a *deferred strategy* for updating objects derivable from base relations. Collectively, we refer to objects, such as views, secondary indexes, downloaded relations or views, and multiple copies, as *derived objects*. Maintenance of derived objects is deferred until a direct or indirect request is made to them. This avoids overhead associated with update broadcast. When a query needs to access a possibly outdated view, say V , the incremental update algorithms propagate changes of the base relations to all intermediate views used in the derivation of V . The update cost of V is part of the cost of accessing V . Since the incremental update plus the cache cost is less than the cost of reexecuting the definition of the view, maintenance of views is free of overhead.

The same technique is used with secondary indexes. When a query needs to use an outdated index, it updates it first. This is done by batching, sorting, and merging all the updates together. The cost

of maintaining a given index now becomes the cost of using it in answering the query instead of a uniformly distributed overhead cost.

In the ADMS $\pm$ environment, the deferred update strategy is even more important because of the large distribution and multiplicity of derived data objects. The system is not burdened by the overhead of broadcasting every change to every workstation that has data objects affected by the change. This keeps the communication control overhead low—even when the mainframe serves hundreds of workstations, each of which maintains hundreds of downloaded objects. The other significant benefit of the deferred strategy is the efficiency of batching and processing the updates when needed instead of one at a time.

The deferred update strategy also permits defining a new class of concurrency control protocols for redundantly derived or replicated data.

Slowdown for speeding up: a fully concurrent protocol. ADMS uses a concurrency control protocol based on a new type of lock, called *derived object lock*, or *d-lock* for short. It allows concurrent retrieval and update of access paths and other derived objects. Because of the deferred update strategy, a query involving retrieval of derived objects on a given access path is not necessarily a pure retrieval process. Very often, while retrieving from an access path, the access path update algorithm must be invoked to update any outdated subpaths so that we can retrieve from them.

To retrieve/update a derived object on an access path, a d-lock on it is acquired. A d-lock is preceded by a shared lock to all base relations deriving the object. This implies that while an access path's derived object is being brought up-to-date, concurrent access to the deriving base relations is allowed but updates are not. Nor are dependent paths allowed to retrieve/update concurrently. However, as the retrieve/update of an access path proceeds, previously dependent subpaths are released from their d-locks as they become independent, and concurrent retrieval/update through them is then permitted.

The simple example in Figure 2 shows two paths, *who.u253* and *u253.what*, that have a common subpath, *u253*. The first path to reach *u253* requests a d-lock on it. If this request is granted to *who.u253*, for example, the d-lock will force path *u253.what* to wait at *u253*. As soon as *u253* is updated, its d-lock is released, and path *u253.what* can run in parallel with *who.u253*.

The most exciting aspect of this protocol is that it is a fully concurrent, non-delay protocol. Even when a process B

requesting an update—say of path $u253$, what as in the previous example—waits for another process A to finish updating a dependent subpath, the waiting process B will not have to update the common subpath, $u253$ and above. The cost of updating shared paths is paid only once by the d-lock holder. Since the earlier process A had d-locked the common subpath, the waiting process B has to wait only for A 's remaining time, which is a fraction of the time it would take B to do it by itself. Thus, the more a process is slowed down by finding more and more d-locked objects, the speedier its execution! Note that, since almost zero time is needed to check whether an object is up to date, the slowdown-for-speeding-up concept holds for any number of waiting processes.

The ADMS+ separation of access paths to global and local allows this architecture to provide two levels of concurrency and security control. For global paths processed on the mainframe, the d-lock-based protocol for derived objects (a standard concurrency protocol using exclusive locks¹⁰ for updates on base relations) and a security protocol¹¹ are used. When processing of the global subpath is finished, all locks are released before the processing of the local subpath begins. No concurrency control is needed on the workstation because all updates to base relations are transmitted to and processed at the mainframe. No security control is needed because new access paths are checked once at the mainframe before downloading. If they get downloaded, they remain accessible until their global security status changes.

The transfer of control from global to local results in an early release of locks that significantly increases the mainframe concurrency compared to the concurrency achieved when the whole access path resides on the mainframe. Furthermore, since local access paths on a workstation are mostly independent of local access paths on other workstations and since no synchronization-delay overhead is incurred, overall concurrency is even higher than that of a fully distributed and non-redundant architecture.

Access path distribution protocols

In ADMS+ two protocols distribute access paths between the mainframe and workstations. The two protocols differ in the bindings they permit between mainframe and workstation objects.

The access path distribution (APD) protocols are designed to

(1) Maximize localization of subsets of data and access paths tailored to the ap-

plications of a few workstations, but not general enough to be supported at the global level.

(2) Maximize globalization of those access paths common to a good number of workstations. A common access path is considered global, and thus uploaded to the mainframe, if it is shared by k or more workstations, where k is set up by a policy maker. Since it is accessed more frequently, a global path is more apt to be up to date. If updating is required, the cost paid in response to a requesting workstation can be depreciated against subsequent requests.

(3) Keep target relations of all access paths of a workstation local. Keeping target relations on the workstation, even when they have global counterparts on the mainframe, provides very rapid access on the workstation.

(4) Minimize the overhead of broadcasting promotions or demotions of access paths from local to global or vice versa.

A brief explanation of ADMS+ catalog management is necessary here. The ADMS+ catalog tracks not only global objects but also local objects on any workstation that are bound to global objects. The ADMS- catalog tracks all global objects and objects local to the workstation but has no knowledge of objects local to other workstations. From its catalog, ADMS- cannot tell whether the creation of a new access path makes this path promotable to global, or whether a removal demotes it to local. But this is recognized by ADMS+, which tracks the commonality of access paths. When k is reached by the creation of a new path, the requesting workstation is notified about the promotion in order to upload it. When $k-1$ is reached by a removal of a global path, the demotion notification is sent to the requesting workstation for its removal from the local catalog. Otherwise, it stays global but is no longer considered local to the removing workstation. Path promotion or demotion is never broadcast to other workstations sharing the path. They will be notified next time they request an access that has, in the meantime, been promoted/demoted.

The access path distribution protocols are best described in terms of query language requests. ADMS+ query requests that affect the distribution of access paths can be classified into four types:

- (1) creation of a new global base relation,
- (2) display of an existing global base relation or a global view,
- (3) derivation of a new view from an access path that consists of a set of existing or new intermediate base relations and/or views, and

- (4) removal of a global base relation, a global view, or a local view.

Each type of request is executed according to the rules of the protocol.

APD1—static APD protocol. The static access-path-distribution protocol allows only identity bindings between mainframe and workstation objects; thus, the incremental update algorithms of this binding have to be used. Because of the simplicity of the identity binding, a single execution choice is possible. Thus, APD1 is a static protocol. Using APD1, every access path in the mainframe is shared by at least k workstations. We now examine how each query type request is handled by APD1.

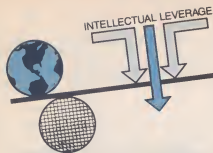
Creation of a new base relation. Every base relation is created on the mainframe. No copy of the relation is retained on the requesting workstation and, therefore, no binding is created. (We assume that the creation of private base relations is handled by the workstations locally and that the mainframe and the rest of the workstations never become aware of their existence.)

Display of an existing global relation or a global view. When a global relation or a global view is requested for display, it becomes a target relation. To meet the third APD design principle, it is downloaded, and the identity binding is formed between the global and the target relation.

Derivation of a new view. All mainframe relations required for the derivation but not on the workstation are downloaded. As a result, appropriate identity bindings are formed. The rest of the access path is derived locally on the workstation.

If constructing a new local view V makes a workstation the k th one sharing V , V and all local parts of its derivation path must be uploaded and become global. However, for each view V' on this local path that is either a target relation or used in the derivation of a third view, its $M(V')$ is retained on the workstation and identity binding between the global V' and $M(V')$ is formed. These upload rules are dictated by the third APD principle of keeping target relations on the workstation and by the identity binding requirement that any derivation from a global object V' be preceded by a download of $M(V')$ followed by local derivations from $M(V')$.

Removal of a global base relation, global view, or local view. Removal of a global relation or a view affects the access path deriving it and all other access paths related to it. Bindings are also either directly or indirectly affected.



FOR YOUR PROFESSIONAL GROWTH **Attend COMPCON 87**

The year's only *broad-based* computing conference
sponsored by IEEE-CS, the world's largest computer society.

Preliminary Program Data: 39 Sessions

Plenary Session Presentations

Future Directions In High Performance Work
Stations. Prof. Andries Van Dam, Brown Univ.
Trends In Knowledge Processing: From Expert
Systems To Intelligent Systems Engineering.
Dr. Frederick Hayes-Roth, Teknowledge Inc.

Networks, Local & Open Systems

Debate, Correct Interface to File Servers
Local Area Networks
ISO—OSI Pros & Cons

AI, Architecture & Expert Systems

The Xenologic X1
Expert System Development Environments
Intelligent Systems for Management Decisions
Manufacturing AI: CAD/CAM, Prod. Eng.
Knowledge Net
Expert System, Tools and Applications
Neural Based Architectures: Breakthroughs Real?

Supercomputers and Superminis

Supercomputers: ETA 10, IBM 3090, Eklx.
Warp: Systolic Supercomputer
Second Generation Superminis

Security, Trusted Networks

Trusted Systems
Trusted Networks
Computer Security

Software, Languages & Systems

LISP Benchmarks
Future Computing Environments
COBOL 85—The New Productivity Tool
Distributed Operating Systems
Technical Directions of UNIX
Multimedia Information Environments
High Performance Transaction Systems
Managing Change In High Availability Systems

Software, Technologies

Object Oriented Models
Software Reuse—The State of the Malpractice
What's Up with Ada Compiler Technology?
Reusable Software Component Libraries
Window Systems, Diverging Views of Future

Interesting Computer Organization

AT&T Bell Labs C Machine
Alternative Computer Architectures
VAX 8800 System
Audience Open Mike, Your Provocative Proposals

Optics, Computing & Storage

Optical Disks, Present and Future
New Directions in Optical Computing

CAD, Parallel Processing

Parallel Processing for CAD Applications

Digital Signal Processing

Digital Signal Processing

Conference Tuesday Feb. 23 through Thursday Feb. 27, 1987
Cathedral Hill Hotel, San Francisco, California

See January 87 COMPUTER MAGAZINE for Advance Program and registration forms.

COMPCON Tutorials

Monday, Feb. 23, 1987

"Local Networks—Changing Directions." Harvey Freeman
"Secure Operating Systems." Lyle (Terry) Cox
"AI Machines." David Shaw
"Software Reuse: State of the Practice." Will Tracz

Friday, Feb. 27, 1987

"Reusable Software Engineering." Peter Freeman
"Practical Computer Security & Cryptography" Russell Brand & Gilles Brassard
"AI Managing Knowledge Systems Software Development." Avron Barr

For More Information Contact:

Program Information: Glen G. Langdon, Jr., Publicity Chairman
IBM Dept. K54/802 650 Harry Road, San Jose, CA 95120-6099 (408) 927-1818
Registration Information: Lori Gorz, COMPCON Administrator
LLNL P.O. BOX 808, L-130, LIVERMORE, CA 94550 (415) 422-8934

SPONSORED BY THE
 IEEE COMPUTER SOCIETY

THE INSTITUTE OF ELECTRICAL AND
ELECTRONICS ENGINEERS, INC.

A request for removing a global base relation R causes the removal of R from the mainframe and of $M(R)$ from the workstation. All global (local) views derived from $M(R)$ are also removed and their bindings are broken.

When the removal of a view V is requested, it is removed from the workstation only. If it is bound to the mainframe object, the corresponding binding is broken. All views, global or local, derived from V are also removed and related bindings are broken. Any binding created merely for the derivation of V is broken. Note that, according to the fourth APD principle, even though the removal of a view V triggers its demotion—breaks the k th identity binding— V remains on the mainframe until all other workstations having bindings to V make a direct or indirect reference to it, at which time these bindings get adjusted.

APD2—dynamic APD protocol. The dynamic access-path-distribution protocol allows derivation bindings as well as identity bindings. In APD2, as in APD1, every mainframe access path is shared by at least k workstations, but both types of bindings are possible between mainframe and workstation objects. For operations where both bindings are applicable, either one can be chosen dynamically, depending on the mainframe's load, speed, and so on. In contrast, APD1 provides a simpler but fixed execution distribution of access paths.

The description of this protocol is the same as that of APD1 except for the derivation of a new view. Therefore, we will only cover that case.

Derivation of a new view. In the derivation binding a view can be derived either on the mainframe, the workstation, or both. In each of these cases, the derivation contains a local subpath which, because of the current request, may become promotable. Then all views of that subpath must be promoted.

When a new view $M(V)$ created by a derivation binding becomes global, then it is uploaded as V to the mainframe. If $M(V)$ is a target relation for the workstation, the derivation binding of V is broken and a new identity binding between the global V and $M(V)$ is created. If, on the other hand, $M(V)$ is not a target relation but is used in the derivation of another target relation V' , then either $M(V)$ can be bound by an identity binding to V , as above, or $M(V')$ is bound to V by a new derivation binding.

Global and local control parameter. The parameter k is used to control globalization. When k gets smaller and smaller, access paths become more and more global,

and workstations have less and less computation to do on their own. This increases mainframe load and reduces the overall system concurrency. On the other hand, when k gets bigger and bigger, access paths become less and less global, and workstations have to do more and more computation on their own. This increases concurrency and distribution of the load. Thus, k provides a spectrum of access path distribution.

Two very interesting distributions are achieved by giving k its two extreme values. When k is one, everything becomes global. This achieves maximum access path sharing at the cost of increased mainframe load and reduced concurrency. The other extreme is when k becomes bigger than the number of workstations. In this case, only global base relations are maintained on the mainframe and all access paths are local. This may be a very desirable approach for a less-capable mainframe that (due to load, speed, storage, and so on) becomes the system bottleneck, or when a policy of confining all load generated by a workstation within itself is desirable. In this extreme case of k , no overhead is incurred other than the absolute minimum required to maintain the database consistent in a multi-user environment. All other processing is independently and concurrently done on the workstations.

In brief, this new database systems architecture fully integrates local and global database management. The implementation of an ADMS+ prototype is near completion. Further exploration of the incremental approach and its principles is already under way. We are applying these principles and techniques presented to fully distributed databases with replicated copies that can be incrementally maintained. □

Acknowledgment

This work was supported in part by NASA under contract NAS-5-29265 and in part by the University of Maryland Institute for Advanced Computer Studies.

References

1. N. Roussopoulos and H. Kang, "Preliminary Design of ADMS+: A Workstation-Mainframe Integrated Architecture for Database Management Systems," *Proc. 12th Int'l Conf. Very Large Databases*, Morgan Kaufmann, Los Altos, Calif., Aug. 1986, pp. 355-364.
2. N. Roussopoulos, C. Bader, and J. O'Connor, "ADMS: An Advanced Database Management System: Design Document," Dept. of Computer Science, Univ. of Maryland, College Park, Jan. 1984.
3. M. Stonebraker, "Implementation of Integrity Constraints and Views by Query Modification," *ACM SIGMOD Int'l Conf. Management of Data*, 1975, pp. 65-78.
4. N. Roussopoulos and H. Kang, "Incremental Update and Cache of Views: Concept and Cost Analysis," Dept. of Computer Science, Univ. of Maryland, Dec. 1985.
5. L. L. Traiger et al., "Transactions and Consistency in Distributed Database Systems," IBM Research Report RJ2555, June 1979.
6. M. Stonebraker, "Concurrency Control and Consistency of Multiple Copies of Data of Distributed INGRES," *IEEE Trans. Software Eng.*, Vol. SE-6, No. 3, May 1979, pp. 186-194.
7. C. J. Date, *An Introduction to Database Systems*, Vol. II, Addison-Wesley, Reading, Mass., 1983.
8. N. Roussopoulos, "View Indexing in Relational Databases," *ACM Trans. Database Systems*, Vol. 7, No. 2, June 1982, pp. 258-290.
9. N. Roussopoulos, "The Logical Access Path Schema of a Database," *IEEE Trans. Software Eng.*, Vol. SE-9, No. 5, Sept. 1982, pp. 563-573.
10. K. P. Eswaran et al., "The Notions of Consistency and Predicate Locks in a Database System," *Comm. ACM*, Vol. 19, No. 11, Nov. 1976, pp. 624-633.
11. N. Roussopoulos and C. Bader, "Dynamic Access Control for Relational Views," *Information Systems*, Vol. 10, No. 3, Aug. 1985, pp. 361-369.

Nick Roussopoulos is the guest editor of this issue of *Computer*. His photo and biographical sketch appear on page 8.



Hyunchul Kang is currently completing the requirements for the PhD degree at the University of Maryland, College Park. His research interests are in database systems, distributed systems, and data structures.

Kang received the BS degree in computer engineering from Seoul National University in 1983 and the MS degree in computer science from the University of Maryland.

The authors' address is Dept. of Computer Science, University of Maryland, College Park, MD, 20742.

Metadata Management



Leo Mark and Nick Roussopoulos
University of Maryland

A self-describing database system uses an active and integrated data dictionary to provide metadata to systems and users. The data dictionary system uses the services of the database system to manage metadata.

A database system should support a rich variety of metadata describing and controlling the management and use of data. Few database systems today provide even rudimentary integrated metadata management facilities.

This article presents a self-describing database system. Its active and integrated data dictionary system provides the only source of metadata to users, programs, and the database system, and uses the services offered by the database system for metadata management. We focus on the design of a self-describing metaschema and a formalism for specifying some of the operations controlling the evolution of database schemata.

Architecture for self-describing database systems

A self-describing database system and its environment are illustrated in Figure 1.^{1,2} The data mapping control system, or DMCS, supports and enforces two orthogonal dimensions of data description, the point-of-view dimension and the intension-extension dimension.

The point-of-view dimension has three levels of data description: the information meaning described in the conceptual schema, the external data representations described in external schemata, and the internal physical data structure layout described in the internal schema. These

three levels of data description result in databases that are flexible and adaptable to changes in the way users view the data and in the way data is stored. This combination of flexibility and adaptability is usually called data-independence.³

The intension-extension dimension has four levels of data description:

- (1) the information about the data model supported by the database system described in the meta-schema,
- (2) the information about the management and use of data described in the data dictionary schema,
- (3) the information about specific applications described in the application schemata, and
- (4) the application data.

Each level of data description in the intension-extension dimension is the intension of the succeeding data description and the extension of the preceding data description. An intension completely describes and controls changes of an extension. All the levels of the intension-extension dimension are described in terms of the same data model. A description of the metaschema is explicitly stored as part of its own extension—it is self-describing. The stored metaschema can be retrieved, but it cannot be changed; it is self-describing, not self-destructing.

The DMCS can be thought of as a DBMS stripped to the bones on the one hand, but still a complete DBMS on the other hand. It supports the set of elementary functions essential to the maintenance of the two dimensions of data description.

The data language (DL) interface is the data manipulation language for the data model. Because the metaschema is self-describing, all data, including descriptions, is defined, retrieved, and manipulated through the DL interface. No data definition language is needed.

The data management tool box contains software that is plug-compatible with the DMCS through the DL interface. A data management tool supports high-level functions that, although important, are not essential to data management. To produce a plug-compatible data management tool, information about the data model must be retrievable through the DL interface. This is exactly why the metaschema is explicitly stored. Each of these tools will have its own user interface, but each must interface with the DMCS through the DL interface. A database administrator would develop or buy off-the-shelf data management tools, such as schema design aids, software documentation packages, high-level query language processors, report generators, etc., to supplement the elementary functions supported by the DMCS. He or she would develop or buy off-the-shelf definitions of the data dictionary schema and definitions of application schemata for classes of applications.

The internal data language (i-DL) interface is the interface through which all data is passed from the DMCS to the operating system supporting the DMCS.

The two orthogonal dimensions of data description supported by the DMCS, the

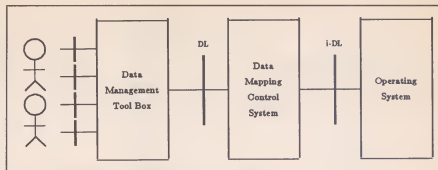


Figure 1. Self-describing database system environment.

point-of-view and the intension-extension dimension, are illustrated in Figure 2.

Any system in this new architecture is born with data structures to hold the metaschema extension—the data dictionary schema. Initially, the data dictionary schema consists of the stored description of the metaschema. The data dictionary schema can subsequently be expanded into a full data dictionary schema describing and controlling the management and use of application databases. Initially, the extension of the data dictionary schema, the data dictionary data, consists of an empty set of tables.

The architecture supports software plug-compatibility and data plug-compatibility, which gives the database administrator the freedom to choose the tools for data management and define the

data management strategy best suited for the enterprise.

The architecture unifies well-established principles and current trends in the areas of database systems and data dictionary systems. It is a generalization of the ANSI/SPARC DBMS Framework, which only considers the point-of-view dimension, and comprises recent ideas on the intension-extension dimension from the International Standards Organization Working Group (ISO/TC97/SC5/WG3) on the conceptual schema and information base. It supplements data model standardization efforts and supplements data dictionary system standardization efforts.

The architecture is based on the notion of self-describing database systems and has matured through several discussions in the ANSI/SPARC Database Architecture

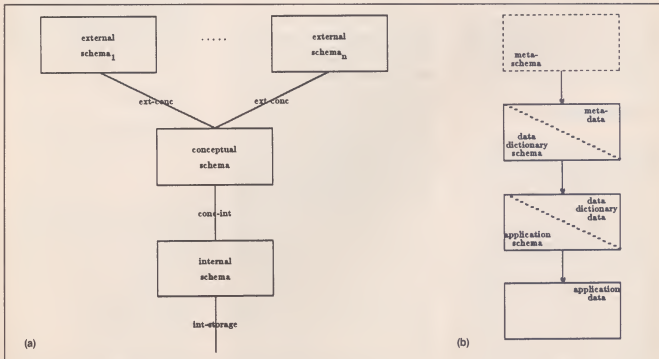


Figure 2. Point-of-view dimension (a) and intension-extension dimension (b).

Figure 3. Graphic formalism for data structures.

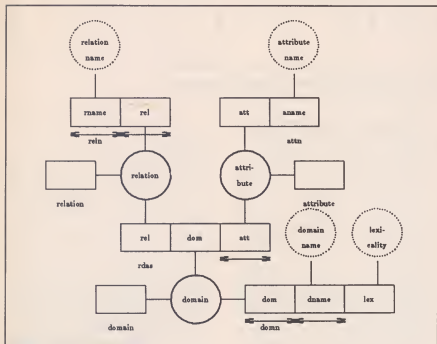
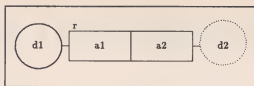


Figure 4. Core metaschema.

Framework Task Group, DAFTG. The architecture has been accepted by ANSI/SPARC and is being considered by ISO as a reference model for database systems in the late 1980's and the 1990's.

The metaschema

We use the relational data model as an example.⁴ However, we use a graphic formalism inspired by the object-role data models.^{5,6} The symbols in Figure 3 represent the definition of a relation with name *r* and two attributes with names *a1* and *a2*. The attributes are defined over a nonlexical domain with name *d1* and a lexical domain with name *d2*, respectively.^{7,8}

The entities in relational schemata are definitions of relations, domains, and attributes. The entity names used in these definitions are relation names, domain names, and attribute names. Accordingly, we define in Figure 4 the nonlexical domains named relation, domain, and attribute, and the lexical domains named relation-name, domain-name, and attribute-name. The three unary relations

named relation, domain, and attribute define sets of entities in existence. The relation named *reln* defines relationships between existing relations and their relation-name. The relation named *attn* defines relationships between attributes and attribute-names. The relation named *domn* defines relationships between domains and their domain-names and lexicality. Finally, the relation named *rdas* defines the relationships between relations, domains, and attributes. The keys, indicated by double headed arrows, point out attributes, the values of which uniquely identify tuples in the relation. Note that no two relations can have the same name and no relation has more than one name. The same restriction applies to domains and their names. In *rdas* we indicate that an attribute is a unique entity related to at most one domain and relation.

There are several more constraints involved: the relations relation, domain, and attribute model sets of entities in existence. Therefore, $rein[relation] \subseteq relation[relation]$ and corresponding rules apply for domains and attributes. On the other hand, we insist that all relations have

names, so actually $rein[relation] = relation[relation]$. Also, attribute names must be unique within relations.

These rules, together with the keys and several other rules, will all be specified as part of the operations defined in the metaschema.

Boxes represent metaschema relations. Full circles represent nonlexical domains of surrogates used to model entities. Broken circles represent lexical domains used to model entity-names. Arrows represent keys. The metaschema is so far self-describing. It is defined in terms of relations, domains, and attributes, and its definition can be stored in the database it defines. See Figure 5. (We have omitted the unary relations relation, domain, and attribute.)

Operations in the core metaschema

To make sure that the metaschema completely models and controls all operations on its extension (the data dictionary schema), we define an insert, delete, and modify operation for each relation in the metaschema. From the database administrator's point of view these operations are elementary, but as we shall see in their specification below, each involves several implied operations on update-dependent relations. All the operations we define must be explicitly represented in the metaschema, otherwise the metaschema does not completely model and control all operations on its extension. For the metaschema to be self-describing, it must explicitly model the notion of an operation and control all operations on the specification of operations. We do not consider this expansion of the core metaschema here. For a detailed description, see Mark.⁹ Operations are specified in the language we now describe.¹⁰

Syntax. Each operation is defined by a set of update dependencies, each with the following form:

$\langle op1 \rangle \rightarrow \langle cond \rangle, \langle op2 \rangle, \dots, \langle opn \rangle$.

where $\langle op1 \rangle$ is the operation being defined; $\langle op1 \rangle, i = 2, \dots, n$, is either an implied operation or an implied primitive operation; and $\langle cond \rangle$ is a condition on the database state.

An operation $\langle op1 \rangle$ has one of the following forms:

```
insert(<relation_name>
  (<tuple_spec>))
delete(<relation_name>
  (<tuple_spec>))
modify(<relation_name>
  (<tuple_spec>)),
<relation_name>(<tuple_
  spec>))
```

reln		domn			rdas			attn	
rname: relation name	rel: relation	dom: domain	dname: domain name	lex: lexicity	reln: relation	dom: domain	att: attribute	att: attribute	aname: attribute name
reln	r1	d1	relation	non-lexical	r1	d4	a1	a1	rname
domn	r2	d2	attribute	non-lexical	r1	d1	a2	a2	rel
attn	r3	d3	domain	non-lexical	r2	d3	a8	a3	att
rdas	r4	d4	relation	lexical	r2	d6	a9	a4	aname
		d5	name	lexical	r3	d2	a3	a5	rel
		d6	attribute	lexical	r3	d5	a4	a6	dom
		d7	name	lexical	r4	d1	a5	a7	att
			domain	lexical	r4	d3	a6	a8	dom
			name	lexical	r4	d2	a7	a9	dname
			lexicity	lexical				a10	lex

Figure 5. Metaschema description stored in metaschema extension.

where the $\langle \text{tuple spec} \rangle$ is a tuple variable for the relation with the name $\langle \text{relation name} \rangle$ and consists of a list of $\langle \text{domain variable} \rangle$ s. The $\langle \text{tuple spec} \rangle$ in $\langle \text{opl} \rangle$ is the formal parameter for $\langle \text{opl} \rangle$. All the $\langle \text{domain variable} \rangle$ s in the $\langle \text{tuple spec} \rangle$ of $\langle \text{opl} \rangle$ are assumed to be universally quantified. All $\langle \text{domain variable} \rangle$ s in the $\langle \text{tuple spec} \rangle$ of $\langle \text{opl} \rangle$, not bound to a universally quantified $\langle \text{domain variable} \rangle$ in $\langle \text{opl} \rangle$, are assumed to be existentially quantified. All $\langle \text{domain variable} \rangle$ s are in caps; nothing else is.

The implied primitive operations are assert for adding a new tuple in a relation, retract for eliminating one, write and read for retrieving data from the user, new for creating a unique new surrogate, and break for temporarily stopping the system to do some retrieval before giving control back to the system. The implied primitive operations $\langle \text{opl} \rangle$ have the following forms:

```

assert(<relation name>(<tuple
spec>))
retract(<relation name>(<tuple
spec>))
write("<any text>"), or write
(<domain variable>)
read(<domain variable>)
new(<relation name>
(<tuple spec>))
break

```

The $\langle \text{relation name} \rangle$ used in the operation new must be the name of a unary relation defined over a nonlexical domain. The conditions $\langle \text{cond} \rangle$ are expressions of predicates with the form $\langle \text{relation name} \rangle(\langle \text{tuple spec} \rangle)$. The connectives used in forming the expressions are $\wedge$ (and) and $\neg$ (negation). In addition, we use the primitive predicates nonvar and var to decide whether or not a $\langle \text{domain variable} \rangle$ has been instantiated.

Conditions can also be used by the user to retrieve data from the system.

Semantics. An operation succeeds if, for at least one of its update dependencies, the condition evaluates to true and all the implied operations succeed. It fails otherwise.

When an operation is invoked, its formal parameters are bound to the actual parameters. The scope of a variable is one update dependency. Existentially quantified variables are bound to values selected by the database system or to values supplied by the user upon request from the database system. Evaluation of conditions, replacement of implied operations, and execution of implied primitive operations are left-to-right and depth-first for each invoked update dependency. For the evaluation of conditions we assume a closed world interpretation.¹¹

The nondeterministic choice of a replacement for an implied operation is done by backtracking, selecting in order of appearance the update dependencies with matching left-hand sides. If no match is found, the operation fails.

An implied operation matches the left-hand side of an update dependency if

- the operation names are the same, and
- the relation names are the same, and
- all the domain components match.

Domain components match if they are the same constant or if one of both of them is a variable. If a variable matches a constant, it is instantiated to that value. If two variables match, they share value.

The semantics of the primitive operations are

assert($r(t)$)

Its effect is $r := r \cup \{t\}$. It always succeeds. All components of t are constants.

retract($r(t)$)

Its effect is $r := r \setminus \{t\}$ where all components of t are constants. It always succeeds.

write("text")

It writes the "text" on the user's screen. It always succeeds.

read(X)

It writes the value of X on the user's screen. It always succeeds.

read(X)

It reads the value supplied by the user and binds it to X . It always succeeds (if the user answers).

new($r(D)$)

It produces a new unique surrogate⁷ from the nonlexical domain over which r is defined and binds the value of the variable D to this surrogate. It always succeeds.

break

It suspends the current execution and makes a new copy of the interpreter available to the user, who can use it to retrieve the information needed to answer a question from an operation.

We kept the list of primitive operations minimal to illustrate the concept. It can easily be extended. We should emphasize that the user cannot directly invoke primitive operations.

The execution of assert and retract operations done by the system in an attempt to make an operation succeed will be undone in reverse order during backtracking. This implies that an operation that fails will leave the database unchanged.

The operations. We will only specify the few operations shown in the table in Figure 6 to illustrate the principles. For a detailed discussion, see Mark.⁹

rel/op	insert	delete	modify
domain			
domn			
relation	x	x	
reln	x	x	
attribute	x	x	
attn			
rdas	x	x	

Figure 6. The operations.

Figure 7.
Insertion into relation.

```

insert(reln(R))
→ var(R),
  new(reln(R)),
  insert(reln(R)).
→ nonvar(R) ∧ reln(R),
  nonvar(R) ∧ ¬(reln(R)),
  assert(reln(R)),
  insert(rdas(R, _)),
  insert(reln(_R)).

```

Figure 8.
Insertion into reln.

```

insert(reln(N,R))
→ var(R),
  new(reln(R)),
  insert(reln(N,R)).
→ var(N),
  write("relation name?"),
  break,
  read(N),
  insert(reln(N,R)).
→ nonvar(N) ∧ nonvar(R) ∧ reln(N,R),
  nonvar(N) ∧ nonvar(R) ∧ ¬(reln(N, _)) ∧ ¬(reln(_R)),
  insert(reln(N,R)),
  insert(reln(R)).

```

```

insert(attribute(A))
→ var(A),
  new(attribute(A)),
  insert(attribute(A)).
→ nonvar(A) ∧ attribute(A),
  nonvar(A) ∧ ¬(attribute(A)),
  assert(attribute(A)),
  insert(attn(A, _)),
  insert(rdas(_A, _)).

```

Figure 9. Insertion into attribute.

If the variable R in the operation $\text{insert}(\text{relation}(R))$ in Figure 7 is uninstantiated when the insertion into relation reln is called, the system produces a new surrogate and proceeds with the insertion. If R is instantiated, two possibilities exist: R is already in relation reln , in which case the insertion succeeds with the database state unchanged; or R is not in relation reln , in which case we insert it. Since all relations must have some attributes and a name, we propagate by triggering insertions into the relations rdas and reln of the tuples $(R, \_)$ and $(\_, R)$, respectively, indicating that the relation surrogate is at this point the only thing we know.

In the operation $\text{insert}(\text{reln}(N, R))$, in Figure 8, the first two rules produce or request from the user any uninstantiated variable values. The insertion operation succeeds with the database state unchanged if a relation with that particular name is already in the database. If no rela-

tion with the name N exists and the relation represented by surrogate R does not already have a name, we assert the tuple and propagate by inserting R in relation reln .

In the operation $\text{insert}(\text{attribute}(A))$ in Figure 9, the first update dependency produces a new attribute surrogate if needed. The second succeeds if the surrogate is already present. The third makes the assertion and propagates through insertions in attn and rdas .

In the operation $\text{insert}(\text{rdas}(R, D, A))$ in Figure 10, a new attribute surrogate is produced if needed. If the attribute is not already in rdas and relation and domain surrogates are not provided, they are requested. The relation name uniqueness within relations is not about to be violated, then the tuple is asserted in rdas . This may cause propagation to relations relation , domain , and finally attribute .

The relation rdas is central to the core metaschema. Since we want domains to exist without being currently used in any relations, the only direct propagation from deleting tuples from rdas is to attribute and sometimes to relation .

The definition of the compound update operation $\text{delete}(\text{rdas}(R, D, A))$ in Figure 11 is very special, because it allows the user free use of any combination of uninstantiated variables, possibly resulting in multiple tuple deletions. The multiple deletions result from extensive use of recursion, and every single deletion is automatically propagated during the process.

If no tuples matching the actual parameter are present in rdas , the operation suc-

ceeds with the database unchanged. If all variables are uninstantiated, we don't allow any change. The operation succeeds with nothing done to the database. The possible combination of instantiated/uninstantiated parameter combinations left are illustrated in Figure 12.

Because the attribute uniquely identifies one tuple, one (domain, relation), all operations with an attribute surrogate cause one tuple deletion from rdas , which is propagated to attribute . If the attribute to be deleted is the last one in a relation, then the deletion also propagates to relation . If only the domain surrogate is given, $\text{rdas}(\_, d, \_)$, then we must delete all uses of that domain in any relation. If only the relation surrogate is given, $\text{rdas}(r, \_, \_)$, then all attributes for that relation are deleted from rdas , implying of course the deletion of the relation too. Finally, if both relation and domain surrogates are given, $\text{rdas}(r, d, \_)$, then we must delete all uses of the given domain in the given relation.

If the attribute is not present in attribute , the operation $\text{delete}(\text{attribute}(A))$ succeeds with the database unchanged. See Figure 13. In the last update dependency, if no value is given, the user is prompted for one and the operation is tried again. In the second update dependency, if a value is given and that value is in attribute , we remove it and propagate by deleting all its names from attn and all its uses from rdas .

It takes only simple arguments to see that if a deletion of an attribute is caused by a previous deletion from rdas , then the propagation to rdas from this update dependency immediately returns with success, because that attribute is not used in rdas anymore (by the very first update dependency).

If, on the other hand, this deletion from attribute is the original one, the propagation back to attribute following the propagation back to rdas will succeed by the first update dependency.

The operation $\text{delete}(\text{relation}(R))$ is shown in Figure 14. If a surrogate for the relation is given and the relation exists, it is removed, its name is deleted, and all attribute and domain relationships to the relation are deleted from rdas .

In the operation $\text{delete}(\text{reln}(N, A))$ in Figure 15, only the relation surrogate or the name is needed to uniquely identify a tuple to be deleted, so there is no internal recursion in this rule. The propagation of the deletion to relation succeeds with the job done, and the returning call of a deletion from reln stops on the first rule.

In a fully expanded metaschema there will be several more relations and operations modeling and controlling all operations on the specification of operations.⁹

```

insert(rdas(R,D,A))
→ nonvar(A) ∧ rdas(←,A).
→ var(A),
  new(attribute(A)),
  insert(rdas(R,D,A)),
→ var(R) ∧ ¬(nonvar(A) ∧ rdas(←,A)),
  write("relation surrogate?"),
  break, read(R),
  insert(rdas(R,D,A)),
→ var(D) ∧ ¬(nonvar(A) ∧ rdas(←,A)),
  write("domain surrogate?"),
  break, read(D),
  insert(rdas(R,D,A)),
→ nonvar(A) ∧ nonvar(R) ∧ nonvar(D) ∧ ¬(rdas(←,A)) ∧
  ¬(rdas(R,←) ∧ attn(A,N) ∧ attn(B,N) ∧ ¬(A=B)),
  insert(relation(R)),
  insert(rdas(R,D,A)),
  insert(relation(R)),
  insert(domain(D)),
  insert(attribute(A)).

```

Figure 10. Insertion into rdas.

```

delete(rdas(R,D,A))
→ ¬(rdas(R,D,A)).
→ var(A) ∧ var(D) ∧ var(R),
  write("nothing done").
→ nonvar(A) ∧ rdas(R,D,A) ∧ rdas(R,←) ∧ ¬(A=B),
  retract(rdas(R,D,A)),
  delete(attribute(A)).
→ nonvar(A) ∧ rdas(R,D,A) ∧ ¬(rdas(R,←) ∧ ¬(A=B)),
  retract(rdas(R,D,A)),
  delete(attribute(A)),
  delete(relation(R)).
→ nonvar(D) ∧ var(A) ∧ var(R) ∧ rdas(R,D,A),
  delete(rdas(R,D,A)),
  delete(rdas(←,D)).
→ nonvar(R) ∧ var(A) ∧ var(D) ∧ rdas(R,D,A),
  delete(rdas(R,D,A)),
  delete(rdas(R,←)).
→ nonvar(R) ∧ nonvar(D) ∧ var(A) ∧ rdas(R,D,A),
  delete(rdas(R,D,A)),
  delete(rdas(R,D,←)).

```

Figure 11. Deletion from rdas.

rdas	(R, D, A)
-	- a
r	- a
-	d a
r	d a
-	d -
r	- -
r	d -

Figure 12. Possible instantiated/uninstantiated parameter combinations.

```

delete(attribute(A))
→ ¬(attribute(A)).
→ nonvar(A) ∧ attribute(A),
  retract(attribute(A)),
  delete(rdas(←,A)),
  delete(attn(A,←)).
→ var(A),
  write("attribute surrogate?"),
  break,
  read(A),
  delete(attribute(A)).

```

Figure 13. Deletion from attribute.

```

delete(relation(R))
→ ¬(relation(R)).
→ var(R),
  write("relation surrogate?"),
  break,
  read(R),
  delete(relation(R)).
→ nonvar(R) ∧ relation(R),
  retract(relation(R)),
  delete(reln(←,R)),
  delete(rdas(R,←)).

```

Figure 14. Deletion from relation.

Farther down the intension-extension dimension

We need to control schema definition from both the metaschema level and the data dictionary schema level. The first question is therefore what the initial content of the intension-extension dimension is. The second question is how we set up the initial content of the intension-extension dimension.

As we shall see, the initial content of the data dictionary schema includes only the bare minimum needed to control the definition and change of application schemata. A data dictionary schema must do much more than that. The third question therefore is how the database administrator chooses, defines, and enforces a database management strategy.

The relations and operations in the metaschema define and control operations only on the immediate extension of the metaschema. They define and control only intralevel propagation of a schema change. Insofar as general rules and laws

can be identified, which is difficult, the metaschema and the data dictionary schema should also define and control interlevel propagation caused by changing an extension interpreted as intension for the next level. The fourth question therefore is how we specify and enforce interlevel propagation.

These four questions will be addressed in the following short sections.

Initial content. We need to control the definition and change of relations and operations from both the metaschema and the data dictionary schema. In the first case, we must control the definition and change of the data dictionary schema. In the second case, we must control the definition and change of the application schemata.

The relations and operations defined in the previous section are not level specific and we can use them at any level where we need to control the definition and change of schemata at the next level. We therefore choose to include the relations and operations defined previously in both the metaschema and the data dictionary

schema, also in accordance with our principle of self-description.

On the other hand, we do need to be level specific when we call and execute an operation on some relation. We choose to identify the level of data description at which a level-specific operation is defined by prefixing operation names with an m for metaschema, a dd for data dictionary schema, and an s for application schema.

A set of level-specific metaschema operations can be defined as shown in

```

delete(reln(N,A))
→ ¬(reln(N,R)).
→ var(N) ∧ var(R),
  write("relation name?"),
  break,
  read(N),
  delete(reln(N,←)).
→ ¬(var(N) ∧ var(R)) ∧ reln(N,R),
  delete(reln(N,R)),
  delete(relation(R)).

```

Figure 15. Deletion from reln.

```

m_delete(reln(N,R))
→ c1,
   delete(reln(N,R)).

```

Figure 16. Metaschema level-specific deletion from reln.

```

m_insert(reln(N,R))
→ c2,
   insert(reln(N,R)).
..
..
..

```

Figure 17. Metaschema level-specific insertion into reln.

Figures 16 and 17. The purpose of the conditions $c1$, $c2$, ... is to protect from changes that part of the data dictionary schema that has the twofold role of being a stored description of the metaschema and an integral part of the data dictionary schema controlling application schema definition and change.

As a simple example of how the m-operations protect the stored description of the metaschema from modifications, consider the condition $c1$ in the operation specification in Figure 16. It must include the following:

$$c1 = \neg(N = name1) \wedge \neg(N = name2) \wedge \dots \wedge \neg(N = namen),$$

where $name1, \dots, namen$ are the names of the metaschema relations.

We choose to let the initial content of the data dictionary data be an empty set of relations ready to hold the extension of the initial content of the data dictionary schema.

Finally, we choose not to include any initial content of application data.

Our choices for the initial content of the intension-extension dimension can be summarized as in Figure 18.

The metaschema consists of the metaschema relations and the operations defined previously, and level-specific metaschema operations as those defined in Figures 16 and 17. The metaschema itself is imaginary, meaning that nothing is actually stored in the dotted line box. We will later explain how the relation definitions of the metaschema itself are hard-wired in the DL-processor.

A description of the metaschema is explicitly stored in the black box of the data dictionary schema. Nothing in the black box can be changed by operations defined in the metaschema—the metaschema is self-describing, not self-destructing. The conditions in the level-specific metaschema operations prevent changes of the stored description of the metaschema. Nothing but the description of the meta-

schema is part of the initial content of the data dictionary schema.

The initial content of the data dictionary data is an empty set of relation tables ready to store the extension of the initial data dictionary schema.

Setting up the initial intension-extension dimension. In order to explain how we set up the initial content of the intension-extension dimension, we must first make some assumptions about the DL-processor.

The DL processor. The DL processor—the DMCS—has the definition of the metaschema built in. This basically means that it knows the names of the metaschema relations, how they are structured, and in which files their extensions are stored. The DL processor does not have the definition of the metaschema operations built in. Instead, it has a search module that retrieves an operation specification given an operation call. It locates m-operations from the data dictionary schema, dd-operations from the data dictionary schema, and s-operations from the data dictionary data.

The execution module executes non-level-specific operations relative to the level-specific operations that called them. If the level cannot be decided from the call, the operation fails.

The DL processor enforces the operational semantics of the update dependency formalism. As part of this, the specification of primitive operations is built in.

The setup. To set up the initial content of the intension-extension dimension, we replace the search module of the DL processor by a booster module. The booster module searches an externally stored set of level-specific metaschema operation specifications to retrieve an operation specification, given an operation call. This externally stored set of level-specific metaschema operations differs from the set we want to store in one sense: the operations do not have the conditions $c1, c2, \dots, cn$ defined above.

We now issue the series of m-operation calls, resulting in the insertion of the metaschema description as part of the data dictionary schema.

Finally, we replace the booster module by the search module, and we are in business.

Note that an insertion into the relation relation at any level creates an empty table at the next level to hold the extension of the inserted relation definition. This is a general rule for interlevel propagation.

Expanding the data dictionary schema. The data dictionary schema must define and control all operations on data used for

database management. Most importantly, the data dictionary schema must control the definition and change of application schemata, as discussed. But, in addition to that, the data dictionary schema must define and control the notions of authorization, user, schema, program, file, distribution, etc.

As the database administrator decides on a database management strategy, he or she will define relations and operations in the data dictionary schema to enforce this strategy, and the part of the data dictionary schema copied from the metaschema will gradually be expanded into a full data dictionary. This means that the level-specific operations defined in the data dictionary schema will be more complicated than those for the metaschema. As the data dictionary schema expands, the level-specific operations in the data dictionary schema must control the propagation of changes of application schemata to changes of data dictionary data controlled by the expanded data dictionary schema.

In an expanded data dictionary schema, the level-specific operations are constructed from the original metaschema operations, as illustrated in Figure 19.

The conditions $c1, c2, \dots, cn$, can test data controlled by the expansion of the data dictionary schema and thereby distinguish alternatives for propagating changes of data, controlled by the initial data dictionary schema, into changes of data, controlled by the full data dictionary schema. The alternative sequences of implied operations on the data, controlled by the expanded data dictionary schema, are inserted in the operation specification in Figure 19, as indicated by the dots.

Note that the database administrator must use the original non-level-specific insert, delete, and modify operations when defining the level-specific data dictionary schema operations that constitute the user interface for the database designer.

It is important to realize that the only part of the data dictionary schema that cannot be changed in any way through the metaschema is the initial part controlling relation definition and change. This means that the database administrator can design the database management application to suit the particular needs of the enterprise, the same way a database designer designs an ordinary application. If the database administrator prefers a particular kind of authorization procedure, then he or she can include it. Likewise, if the database needs to be distributed, the distribution model can be defined appropriately.

In summary, the database administrator decides on the database management strategy rather than having to suffer with

an inadequate strategy forced by the database system vendor. Or, since the metaschema is explicitly stored, several independent software houses may offer off-the-shelf, plug-compatible database management strategies (that is, data dictionary schema definitions) that the database administrator can choose from. Such plug-compatible data dictionary schema definitions could only be produced by the original vendor if the metaschema was not explicitly described.

Our database system framework supports the notions of plug-compatible data and plug-compatible software. Therefore, we can strip the DMCS to the bones, leaving only the DMCS functions, which are absolutely essential. We know of no other database system or data dictionary system born this naked: they all come with more of the database management strategy built in, such as an authorization strategy, and with built-in data management tools that are nice and important, but not essential.

In summary, the simplicity and potential of our framework is based on

- the explicitly stored metaschema description that gives the plug-compatibility;
- the explicitly stored and changeable data dictionary schema that allows us to design our own database management strategy; and
- the power of the update dependency formalism that allows us to fully follow the 100 percent principle, which means that an intension completely controls its extension and thereby relieves the DL processor from enforcing a lot of special rules.

The following simple example illustrates how the database administrator may expand the data dictionary schema.

Example. The database administrator can define a simple authorization strategy for all application database users by defining the relation and operations in the data dictionary schema using the m-operations, as shown in Figure 20.

When the database designer defines an application schema relation and some operations on it, they will look like Figure 21. (We assume that the DL processor knows the username U.)

When the database designer has defined the operations using the dd-operations, he inserts tuples in the relation authorized, allowing the chosen users to do the chosen operations.

When user u1 calls an operation, such as `s_insert(supplier(.),)`, it will succeed only if the tuple (u1, supplier, s_insert) is stored in the relation authorized.

This was the simple case, where the database administrator trusts that the database designer will remember to insert the condition on the relation authorized in all update dependencies of all s-operations defined. If the database administrator

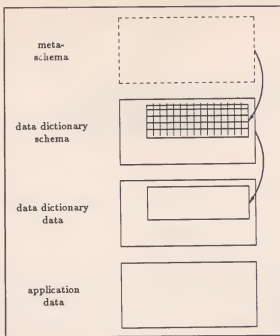


Figure 18. Initial content of the intension-extension dimension.

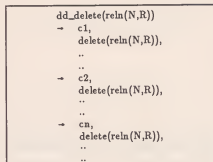


Figure 19. Data dictionary schema level-specific deletion from reln.

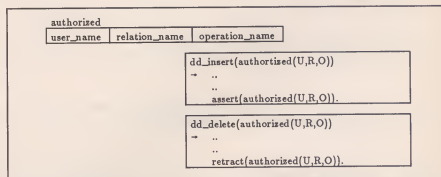


Figure 20. Authorization described in the data dictionary schema.

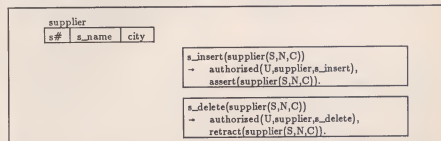


Figure 21. An example of an application schema.

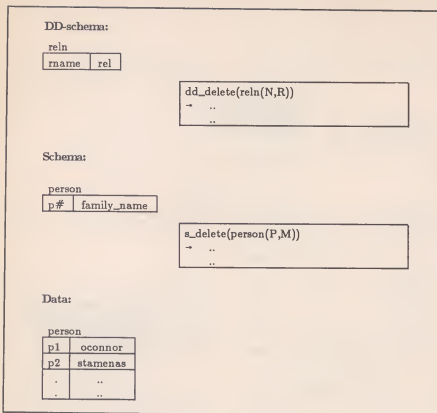


Figure 22. Interlevel propagation.

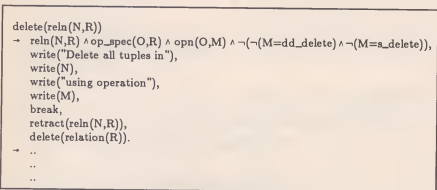


Figure 23. Instructing the user about interlevel propagation.

wants to make sure of this, he or she must define the dd-operations on the relation named condition to force all conditions of s-operations to include authorized (U, <relation_name>, <operation_name>). When the database designer later calls the dd-operations to define s-operations, there is no way to exclude or forget the condition on the relation authorized in any of the update dependencies constituting the s-operations.

Interlevel propagation. We have already given a specification of operations defining and controlling schema updates and

their intralevel propagation. We now study the interlevel propagation of a schema update, that is, the effect a schema update has on the data defined and controlled by the schema.

Example. Consider Figure 22. If we want to delete the tuple (person,r1) from reln, we make the operation call $dd\_delete(reln(person,r1))$. This operation only affects tuples in the database defining the schema. We would somehow like all the tuples in the relation person to be deleted, too.

There are three ways in which we can try

to handle the problem of interlevel propagation. They are all needed. We can

- include a set of rules for interlevel propagation in the semantic definition of the DL;
- specify interlevel propagation directly in the operation specifications; or
- provide a data management tool for database reorganization.

Including interlevel propagation in the semantics of the DL. Only general rules for interlevel propagation for our data model should be included in the semantic definition of the DL.

Before we define these rules, let us illustrate one of the pitfalls of the problem of interlevel propagation by considering the deletion of an attribute from a relation with a nonempty extension. If we think that the general rule for interlevel propagation in the relational model in this situation is enforced by projecting the extension of the relation over the remaining attributes, then we are in for a surprise. The problem is not that we don't know which duplicate tuples, if any, to get rid of. The problem is that the relational projection operator has nothing to do with the process of deleting an attribute from a relation definition. In some situations we may decide that when we delete an attribute from a relation definition, the extension of the new relation should be computed by projection, but this is not a general rule for the relational model.

The reason for our surprise is that we have gotten used to the production of an artificial intension of a relation every time we use the relational operators on some relation extensions. What we must realize is that the relational operators have no intensional semantics. That is, the intension produced by the relational operators has no meaning; it must be assigned by humans.¹²

We know of only a few general rules for interlevel propagation in the relational data model:

- (1) A relation definition can be deleted if the extension of the relation is currently empty. This rule is implemented in Chamberlin¹³ and Zloff.¹⁴
- (2) A relation definition can be inserted. The extension of the relation will be defined to be empty. This rule is implemented in Chamberlin¹³ and Zloff.¹⁴
- (3) An attribute definition can be deleted from a relation definition if the extension of the relation is currently empty. This rule is implemented in Chamberlin¹³ and Zloff.¹⁴
- (4) An attribute definition can be inserted in a relation definition if the extension of the relation is currently empty. This rule is implemented in Zloff.¹⁴ It is generalized in Chamberlin,¹³ where an attri-

bute definition can be inserted in a relation with a nonempty extension. The corresponding values in the extension of the relation are defined to be null, representing value unknown or value inapplicable.

(5) A domain definition can be deleted if it is not part of any relation definition. A domain definition can be deleted from a set of relation definitions if all the attributes defined over the domain can be deleted from the relation definitions. The interlevel propagation is through the deletion of the attribute definitions.

(6) A domain definition can be inserted in a relation definition if the implied attribute definition can be inserted in the relation definition. The interlevel propagation is through the insertion of the attribute definition.

(7) A view definition can be deleted without any interlevel propagation. This rule is implemented in Chamberlin.¹³

(8) A view definition can be inserted without any interlevel propagation. This rule is implemented in Chamberlin.¹³

We include the general rules (1) through (8) in the semantics of the DL. This means that when we insert a base relation definition, the system will create an empty table to hold the extension of the relation, and when we delete a base relation definition, the system will remove the empty table.

All of the above rules for interlevel propagation, except for the generalization of rule (4), actually cover situations where there are no interlevel propagation, except for the empty tables that are set up or removed.

To help the database administrator and the database designer bring the database into a state that allows a schema update, we can let the non-level-specific operations in the metaschema give us a couple of hints, as illustrated in the following example.

Example. In order to enforce a generalization of rule (1), to allow the deletion of a relation with a nonempty extension, we could change the definition of the delete operation on relation *reln* as illustrated in Figure 23.

The condition in the operation checks that there exists a delete operation for the level in question. The relations *op_spec* and *opn* are two of the relations in the expanded metaschema controlling the definition and change of operation specifications. The operation simply tells the user which interlevel propagation must be taken care of before passing control back to the operation.

General rules for interlevel propagation are more interesting in a data model that supports the notion of subtype or is-a relationships. We have defined a set of rules elsewhere.¹⁵

```
dd_delete(reln(N,R))
→ ..
→ ..
→ reln(N,R) ∧ op_spec(O,R) ∧ opn(O,s_delete) ∧ (N=person_name),
  s_delete(person_name(____)),
  retract(reln(person_name,R)),
  dd_delete(reln(person_address,P)),
  dd_delete(relation(R)),
→ ..
→ ..
→ ..
```

Figure 24. Specifying interlevel propagation in an operation.

Specifying interlevel propagation in the operations. Data-dependent rules for interlevel propagation can be explicitly modeled in the operations. If the rules are general for the data model, then they should be included in the non-level-specific operations in the metaschema. If the rules apply to a specific application of the data model, then they should be included in the level-specific operations.

Example. Suppose we define two relations, *person_name*(p#, name) and *person_address*(p#, address), with compound update operations enforcing a referential integrity constraint from *person_address* to *person_name*. This means that when we insert the tuple (pi, address) in *person_address*, then a tuple (pi, name) must be present in or inserted into *person_name*, and vice versa for deletion.

Suppose we want to generalize this rule to the relation definitions themselves, meaning that if we delete the relation *person_name*, then we want to delete the definition of the relation *person_address*, too.

We can specify this rule in the data dictionary schema as shown in Figure 24.

The condition of the operation checks that there exists an operation with the name *s_delete* for the relation to be deleted. The relations *op_spec* and *opn* are two of the relations in the expanded metaschema used to model and control the definition of operations. The operation simply enforces the data-dependent rule that if the relation definition to be deleted is for the relation *person_name*, then all tuples in the extension of this relation must first be deleted.

This technique works very well on data-dependent rules for interlevel propagation. The technique can be used both in level-specific operations in the metaschema and the data dictionary schema.

Database reorganization. When interlevel propagation cannot be included in

the semantics of the DL or explicitly specified in the operations, then we must resort to tools for database reorganization. This means that the database administrator or the database designer will be responsible for the interlevel consistency of intensions and extensions.

Database reorganization is often a very elaborate process involving a system shutdown. Only a few systems, including SystemR¹³ and QBE,¹⁴ support on-line database reorganization. A powerful algebra for database reorganization was proposed for the extended relational model RM/T.¹⁶

On-line database reorganization is supported by a self-describing database system. The general technique follows:

- (1) Insert the definition of the new set of relations.
- (2) Insert the definition of the compound update operations for the new relations.
- (3) Write a data management tool that uses the delete operations on the old relations and the insert operations on the new relations to move and call the data.
- (4) Delete the definitions of the old relations.
- (5) Rename the new relations.

This completes our discussion of the intension-extension dimension. We have described its initial contents and how to set it up. We have explained how to expand the data dictionary schema. And, we have discussed three ways of handling interlevel propagation.

In this article we concentrated on the metadata management aspects of a self-describing database system, and we used the relational data model and the formalism for update dependencies to specify the conceptual level of a self-describing database system.

The architecture for self-describing database systems has been accepted by

ANSI/SPARC and is being considered by the ISO as a reference model for database systems in the late 1980's and 1990's. We will undoubtedly see several database systems develop in this direction.

We are ourselves currently using a self-describing database system in the design of a system for scientific information interchange. □

References

1. L. Mark and N. Roussopoulos, "The New Database Architecture Framework—A Progress Report," *Proc. IFIP WG 8.1 Working Conf. Theoretical and Formal Aspects of Info. Systems*, North-Holland, 1986.
2. T. Burns et al., "Reference Model for DBMS Standardization," Report from the Database Architecture Framework Task Group of the ANSI/X3/SPARC Database System Study Group, Sigmod Records, Mar. 1986.
3. D. Tschritzis and A. Klug (eds.), "The ANSI/X3/SPARC DBMS Framework," *Info. Systems*, Vol. 3, No. 3, 1978.
4. E. F. Codd, "A Relational Model of Data for Large Shared Data Banks," *Comm. ACM*, Vol. 13, No. 6, 1970.
5. B. Breutman, E. Falkenberg, and R. Maur, "CSL: A Language for Defining Conceptual Schemas," *Database Architecture*, G. Brachi and G. M. Nijssen (eds.), North Holland, 1979.
6. G. M. Nijssen, "One, Two or Three Conceptual Schemas," *Database Architecture*, G. Brachi and G. M. Nijssen (eds.), North Holland, 1979.
7. P. Hall, J. Owlett, and S. Todd, "Relations and Entities," *Modelling in Data Base Management Systems*, G. M. Nijssen (ed.), North Holland, 1976.
8. J. J. van Griethuysen (ed.), "Concepts and Terminology for the Conceptual Schema and Information Base," *ISO/TC97/SC5/WG3—N695*, Philips, 1982.
9. L. Mark, "Self-Describing Database Systems—Formalization and Realization," PhD dissertation, TR-1484, Dept. of Computer Science, Univ. of Maryland, College Park, Maryland, 1985.
10. L. Mark and N. Roussopoulos, "Operational Specification of Update Dependencies," submitted to the *ACM Trans. Database Systems*, 1986.
11. R. Reiter, "On Closed-World Data Bases," *Logic and Data Bases*, H. Gallaire and J. Minker (eds.), Plenum Press, 1978.
12. N. Roussopoulos, "Intensional Semantics of a Self-Documenting Relational Model," TR-1264, Dept. of Computer Science, Univ. of Maryland, College Park, Maryland, 1982.
13. D. D. Chamberlin et al., "Sequel 2: A Unified Approach to Data Definition, Manipulation, and Control," *IBM J. Research and Development*, Vol. 20, No. 6, 1976.
14. M. M. Zloff, "Query-by-Example: A Data Base Language," *IBM Systems J.*, Vol. 16, No. 4, 1977.
15. N. Roussopoulos and L. Mark, "Schema Manipulation in Self-Describing and Self-Documenting Data Models," *Int'l J. Computer and Info. Sciences*, Vol. 14, No. 1, 1985.
16. E. F. Codd, "Extending the Database Relational Model to Capture More Meaning," *ACM Trans. Database Systems*, Vol. 4, No. 4, 1979.

Nick Roussopoulos' biography and photograph appear following the Guest Editor's Introduction.



Leo Mark is an assistant professor in the department of computer science at the University of Maryland. His research interests include database management systems architecture, data models, database semantics, database integrity, metadata management, and data dictionary systems.

Mark received his Masters in computer science from Aarhus University, Denmark, in 1980. He first joined the Database Systems Group in the department of computer science at the University of Maryland as a student in January of 1983. There he participated in database architecture standardization efforts for ANSI/SPARC and a research project on a standardized communication environment for NASA. He received his PhD from Aarhus University in 1985.

Readers may write to the authors at the Department of Computer Science, University of Maryland, College Park, MD 20742.

MODULAR LANGUAGES ARE NOT ENOUGH

For the production of high-quality reusable software components, you need all the power of true object-oriented design and programming.

Eiffel™ is the first object-oriented programming system designed exclusively for demanding software designers in practical development environments. Its simple yet powerful design offers a unique combination of features for producing high-quality software:

- classes as the basic system structuring mechanism
- multiple and repeated inheritance
- static type checking and information hiding for security
- assertions and invariants for expressing and checking software correctness
- efficient compilation and execution
- library of highly reusable software components
- automatic configuration management
- automatic run-time storage management, with parallel garbage collection
- supporting tools for documentation and debugging.

Eiffel is available now on most versions of the Unix™ operating system. For other systems, please contact us.

Don't let your software development be hindered by obsolete technology. For reliable, reusable, extensible software, Eiffel makes object-oriented design a reality in today's industrial environments.

For more information on how Eiffel can improve the quality of your software, contact Interactive Software Engineering, Inc. 270 Skokie Road, Suite 7, Grosse Pointe, CA 93117, or call us at (805) 685-1000. We also offer training and consulting in object-oriented methods, and other advanced software engineering tools.

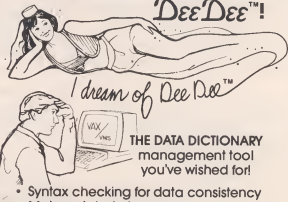


**Interactive
Software Engineering, Inc.**

Professional tools for the development of high-quality software

Eiffel is a trademark of Interactive Software Engineering, Inc. Unix is a trademark of AT&T

If STRUCTURED ANALYSIS is your passion, you'll love DeeDee™!



**THE DATA DICTIONARY
management tool
you've wished for!**

- Syntax checking for data consistency
- Makes global changes
- Alphabetizes automatically
- Menu driven and configurable
- Tailored automated documentation
- Automated creation of sub-definitions
- VAX-like editing of data structures

Keeps LARGE PROJECTS under budget!

Powerful, Productive, Fast!

DeeDee™ ... \$2995

**Ada and Pascal type declarations
created with MkT option ... \$995**

SCS inc., FORT WAYNE, IN

219/432-3975

Views, Objects, and Databases



Gio Wiederhold, Stanford University

Objects provide the useful abstraction in programming languages; views provide a similar abstraction in databases. Since databases provide for persistent and shared data storage, view concepts will avoid problems occurring when persistent objects are to be shared.

The intention of this article is twofold. First of all, I show intrinsic differences in the underlying concepts of access to persistent storage in databases and current extensions of object-oriented programming systems intended to incorporate persistence.^{1,2}

Since the objectives of both paradigms are similar, I develop a connection between *object* concepts in programming languages and *view* concepts in database systems.

Secondly, I propose an architecture that exploits the commonality of objects and views, and indicate research and development directions needed to make the bridge viable. Such an architecture seems to be especially suitable for computer-aided design tasks.

Engineering information systems, or EISs, and systems for similar applications must provide suitable abstractions of real-world objects for their users and support long-lived transactions.³ The complexity of the design process and the number of specialized individuals needed to bring major projects to completion are driving the search for more systematized solutions than those provided by the file mechanisms now in use in operational precursors of EISs. The issues being raised here pertain to systems with large quantities of data and long lifetimes. Design tasks that can be handled by single individuals are not likely to benefit from EIS technology.

Databases and views

Database concepts provide independence of storage and user data formats. The schema describes the form of the database contents, so that a variety of users can satisfy their data requirements by querying the shared database using nonprocedural declarations of the form:

```
SELECT what-I-want WHERE some-  
set-of-conditions-is-true
```

A database administrator integrates the requirements of diverse users. The shared database can be changed as long as the schema is changed to reflect such changes. Concepts of database normalization help avoid redundancy of storage and anomalies that are associated with updates of redundantly stored information.

The principal formal database mechanism to obtain selected data for an application is a view specification. A view on a database consists of a query that defines a suitably limited amount of data. A database administrator is expected to use predefined views as a management tool. Having predefined views simplifies the user's access and can also restrict the user from information that is to be protected. We are interested here only in the first objective, and not in the protection aspects associated with views.

Views have seen formal treatment in relational databases, although subset definitions without structural transformations are common in other commercial database systems. I will hence describe views from a relational point of view, without implying that a relational implementation is best for the underlying EIS databases.

A view is defined in a relational model as a query over the base relations, and perhaps also over other views. Current implementations do not materialize views, but transform user operations on views into operations over the base data. The final result is obtained by interpreting the combination of view definition and user query, using the description of the database stored in the database schema.

The view, like any relational query result, is a relation. However, even when the base database is fully normalized, say to Boyce-Codd normal form, the view relation is, in general, only in first normal form. Views are in that sense already closer to objects: related data has been brought together.

The issue that views present data not in second or third normal form seems ignored in database research, except for the update complications that result.⁴ I have no evidence that the unnormalized views are uncomfortable to a user expecting a normalized relational representation. Acceptance of unnormalized views can be taken as a partial validation of the acceptability of structures more suitable to represent objects than normalized tables. Some current research is addressing unnormalized relations independent from the view aspect.

Objects

Object-oriented programming languages help to manage related data having a complex structure by combining them into objects. An object instance is a collection of data elements and operations that is considered an entity. Objects are typed, and the format and operations of an object instance are inherited from the object prototype.

The prototype description for the object type is predefined and the object instances are instantiated as needed for the particular problem. The object prototype then provides a meta description, similar to a schema provided for a database. That description is, however, fully accessible to the programmer. Internally, an object can have an arbitrary structure, and no user-visible join operations are required to bring data elements of one object instance together.

The object concept covers a range of approaches. One measure is the extent to which messages to external operation interfaces are used to provide access and manipulation functions. Objects may be active, as in the Actors paradigm, or passive, as in CLU, or somewhere in between, as in Simula or Smalltalk in terms of initiating actions.

The use of objects permits the programmer to manipulate data at a higher level of abstraction. Convincing arguments have been made for the use of object-oriented languages and some impressive demonstrations exist. Especially attractive is the display capability associated with objects. Object concepts can of course be implemented using nonspecialized languages, for instance in Lisp or Prolog. The tasks in EIS seem to match object-oriented concepts well and many experiments have been conducted.

Objects and databases

Let us assume a database is used for persistent storage of shared objects. A database query can obtain the information for

an object instance, and an object-oriented programming language can instantiate that object. An interface is needed, since neither can perform the task independently. A view can define the information required for a collection of objects, but the data will not be arranged as expected for a collection of objects. Linkage to the object prototype and its operations is performed in the programming system. The program queries the database nonprocedurally to remain unaffected by database changes made to satisfy other users.

The query needed to instantiate an object may seem quite complex to a programmer. A relation is a set of tuples, and, from an idealized point of view, each tuple provides the data for some object. However, normalization often requires that information concerning one object be distributed over multiple relations, and brought together as needed by join operations. The base relations must contain all the data required to construct the view tuples; the composition knowledge is encoded into the Select expressions used to construct the views. An ideal composition of an object should allow its data to be managed as a unit, but unless non-first-normal form relations—supporting repeating groups—are supported for views, multiple tuples are still required to represent one entity in a relational view.

Hence storage of objects is not easy in databases, as indicated by the extensions proposed to Ingres for such tasks.⁵ A further complexity is that objects themselves may be composed from more primitive objects. In hierarchical databases records may be assembled from lower level tuples, but in relational databases the programmer has to provide join specifications externally to assemble more comprehensive relations from basic relations.

There is some hope that the formal techniques being developed for databases can permit the management of the information required to manage objects. Performance remains a bottleneck, and I will consider this issue later with my proposal. The database community has to be careful not to promise too much too soon to the object-oriented folk.

Sharing information in objects

More serious are the problems I see in the management of shared information within the object-oriented paradigm. I consider two problems:

(1) The growth of objects that contain information for multiple design tasks.

(2) The conflict when object configurations differ for successive design tasks.

Let us first consider the simpler case,

where multiple users deal with the same configuration of objects. I will draw my examples from EIS, and consider that the objects are components of a circuit.

In using an EIS the level of abstraction for the objects changes during the process of design. First the objects are simple logic elements and the process of design refines these objects to circuit components and, eventually, to simple geometric elements projected from each layer of the chip. The final objects will carry many data elements not needed at the higher levels. The sketch in Figure 1 symbolizes how objects grow and lose their vitality.

As the design process moves from one subtask to the next, successor objects are constructed out of earlier objects. Each new generation has new data fields appended.

Unfortunately, since design often requires cycles, old information cannot disappear. An unacceptable geometry may require a change at the circuit level, say adding an inverter to change a polarity. If design is iterative, then successive transformations of objects must not lose information. This means that objects suitable for one task must contain information relevant to all tasks that may be successors, although much information may be irrelevant to the task at hand. The information may be hidden within the object, but must be passed on correctly, so it remains available when needed.

The objects become big, and no longer have the qualities associated with the object-oriented paradigm. A solution to this problem of overloaded objects is to have super-objects, owned by an object czar. Objects for each user task type are created by projection from the super-object. Updating privileges must be well defined.

This solution does not solve the second class of problems, namely sharing object information when the object configuration differs. Now, to create objects for a user task objects may have to be created from combinations of several and, perhaps, different objects or super-objects.

I show in the next section an EIS example having components and nets connecting. These present different configurations of the same information. In aircraft design the objects serve design tasks as aerodynamics, structures, power, fuel, control, etc., and it is obvious that their configuration is quite different. Even in a simple commercial credit card operation there is the customer as an object for some tasks and the establishments are the objects wanting to be paid in other tasks.

Because of these problems, I present later a proposal that will not try to store to objects directly in persistent form. I prefer a new approach to satisfy the demands of database style sharing and object concepts.

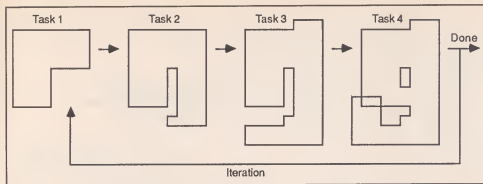


Figure 1. Growth of objects as they try to satisfy multiple design tasks.

Looking at some examples

To clarify the introduction I will take a simple device, a D-type latched flip-flop. At some level of abstraction it is an object; at a lower level it is composed of several gate objects of only three types: AND, INV, NOR, and contacts to the external world. The graphical representation of Figure 2 shows the component objects of the flip-flop at the gate level and the interconnection nets. The components are capitalized and the nets have lowercase labels.

A fully normalized database storing the information describing this circuit requires several relations. I show in Figure 3 the two library relations, which describe the types of gates (Gates) and their connection points (Gate-contacts). Many other values will be kept in such a library. The ruling part or key attributes of each relation are placed ahead of a separator symbol: >.

Figure 4 presents a nonredundant representation for the specific circuit using two more relations, one for each gate instance and one giving the net connections for each gate. Other design-specific information can be kept within these relations.

The representation of the design shown is quite complete, but also fairly unclear. I need to create views that place all relevant data into coherent tuples. A view is needed to analyze the components and their loads; another view is needed to look at the nets; and other views will be needed for timing analysis, heat dissipation, etc.

In Figure 5 I extract two views, ComponentsView and NetsView for the database of Figure 4, using also the library relations shown in Figure 3. The ComponentsView is intended to be appropriate for checking components and their sources and sinks. It primarily accesses the Components relation, and joins with its tuples data about the connected components and from the libraries.

The NetsView is intended to generate data on the interconnection nets for an application doing checking. The primary relation for the NetsView view is the

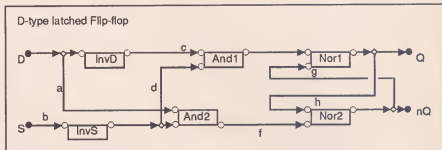


Figure 2. Latched D flip-flop.

Relation	Gates:	Area,	No-pins,	Power,	Delay, . . . ;
Gate-type:	Function,				
Inv	$x = \neg a$	30	2	...	
And	$x = a \wedge b$	40	3	...	
Nor	$x = a \vee b$	35	3	...	
Cntct	$\neg a$	900	1	...	

Relation	Gate-contacts:	C-no ;	C-id,	IO;
Gate-type,				
Inv		1	a	in
Inv		2	x	out
And		1	a	in
And		2	b	in
And		3	x	out
Nor		1	a	in
Nor		2	b	in
Nor		3	x	out
Cntct		1	a	inout

Figure 3. Relations describing the gates library.

Connections relation, augmented with library information for the connected components.

The number of objects in ComponentsView is equal to the cardinality of the component relation (10), but the augmentation makes the result much larger (30). The eight nets are represented in the primary relation and in the view by 20 tuples, one per connected point (o) or external contact (•). In both views I present the tuples in a logical order.

This view is still awkward, since component entities require multiple tuples. A

more reasonable presentation would delete redundancies and add implicit nonfirst-normal-form bindings by rearranging columns according to the source relations. The Nor1 component shown within Figure 5 would then have an object data structure as shown in Figure 6. I add a column N, which gives the number of connected components. The computation of N using SQL requires Group By and Count operations.

I now describe in Figure 7 this structure in the object-oriented extension of C, namely C++ (reference 6). In C++ structures have to be mappable at compile

Relation Components:		
Id,)	Type,	Role,
D	Cntct	Data in
S	Cntct	Sense in
InvD	Inv	Data inverter
InvS	Inv	Sense inverter
And1	And	And data
And2	And	And sense
Nor1	Nor	Nor to Q
Nor2	Nor	Nor to nQ
Q	Cntct	State out
		Position, ...;
		1/1
		3/1
		1/2
		3/2
		1/3
		3/3
		1/4
		3/4
		1/5

Relation Connections:		
Id,	Pin()	net;
D	1	a
S	1	b
InvD	1	a
InvD	2	c
InvS	1	b
InvS	2	d
And1	1	c
And1	2	d
And1	3	e
And2	1	a
And2	2	d
And2	3	f
Nor1	1	e
Nor1	2	g
Nor1	3	h
Nor2	1	h
Nor2	2	f
Nor2	3	g
Q	1	h
nQ	1	g

Figure 4. A fully normalized description.

time, so arrays of dynamic extent cannot be embedded: Since it is permissible in C++ to have a dynamic array as the last element of a class by writing an appropriate constructor, it is possible to bring the CCppin array inboard.

Views and objects

Let us now reconsider the similarities between views and object concepts. Both are intended to provide a better level of abstraction to the user, although the database user is seen to manipulate sets of objects in a nonprocedural notation while the objects are manipulated procedurally and iteratively.

The collection of tuples of a view is defined by the query that generates the set, and described by the relation-schema associated with the view query. The set of objects is defined by the user-initiated action of generation and described by the prototype. Object-oriented languages may have a ForAll primitive to rapidly generate collections of objects of some type.

The description of the relation has to be available to the relational user, since no implied operations can be kept in the relation-schema. There are proposals to store object-defining procedures in relational schemas, but these have not yet been tested.⁵

Both tuples and objects can be selected based on any of multiple criteria, and interrogated to yield data for processing. View update may be severely restricted because of ambiguities in base relation update. Object update can be restricted by having only limited access functions, but otherwise no constraints are imposed on the programmer, although consistency problems can easily arise among users sharing objects.

Basic, of course, is the difference in persistence. Databases, and hence views over them, persist between program invocations. Objects must be explicitly written to files to gain persistence. Related to persistence is the critical issue to be addressed, namely the multiplicity of views that can be derived from a base relation. Figure 5 showed a view of "components" and a

view of "nets" derived from the relations of Figures 3 and 4.

The proposed architectural unit: view-objects

I now propose to combine the concepts of views and objects, as discussed above, into a single concept: view-objects. This proposal is motivated by noting that systems suitable for engineering information problem support require both sharability and complex abstract units, i.e., both views and objects. I use the concept of views to generate a working set of objects corresponding to projections of the entities described in the database.⁷ The generators may need to access multiple relations to reconstruct the entities hidden to the relational representation. Components of the architecture are

- (1) a set of base relations and
- (2) a set of view-object generators.

In a complete version of this approach I also require the inverse function of (2), namely,

- (3) a set of view-object decomposers and archivers.

The conceptual base relations serve as the persistent database for applications under this architecture. They contain all the data needed to create any specified object type. Conventional database technology should be adequate for development, but eventually performance demands may drive users to specialized databases. I will consider the options for physical organization later.

Concurrent access by long-lived transactions will require a capability to conveniently access prior versions of the data. Where database management systems do not serve that need intrinsically, the database must be configured with additional time-based attributes. As experience is gained, this service will be included in specialized systems being built.

The view-object generator is the new component in this architecture. It performs the following steps:

- The view-object generator extracts out of the base database data in relational form as needed—as is now done by a query corresponding to a view. A view tuple or set of view tuples will contain projected data corresponding to an object. A view relation will cover a selected set of objects.

- The view-object generator assembles the data into a set of objects. The objects will be made available to the program by attaching them to the predefined object prototype.

```

CREATE VIEW ComponentsView (ID, Pin, IdC :) Type, N, IO, IOC)
AS SELECT C.Id, C.Pin, CC.Id, CM.Type, T.No-pins, G.IO, GC.IO
FROM Connections C CC, Components CM, Gate-connects G GC, Gates T,
WHERE C.Id = CM.Id AND C.Net = CC.Net AND C.Type = T.Gate-type
AND C.Type = G.Gate-type AND CC.Type = GC.Gate-type;

CREATE VIEW NetsView (Net, Dev, Pd:) IOd;
AS SELECT C.net, C.Id, C.Pin, GC.IO
FROM Connections C, Components CM, Gate-connects G GC, Gates T,
WHERE C.Id = CM.Id AND CM.Type = T.Gate-type;

```

ComponentsView :=				NetsView :=						
Id,	Pin,	IdC :)	Type,	P,	IO,	IOC;	Net,	Dev,	Pd:)	IOD;
D	1	InvD	Cntct	1	inout	in	a	D	1	inout
D	1	And2	Cntct	1	inout	in	a	InvD	1	in
S	1	Inv1	Cntct	1	inout	in	a	And2	1	in
InvD	1	D	Inv	2	in	inout	b	S	1	inout
InvD	2	And1	Inv	2	out	in	b	InvS	1	in
InvS	1	S	Inv	2	in	inout	c	InvD	2	out
InvS	2	And1	Inv	2	out	in	c	And1	1	in
InvS	2	And2	Inv	2	out	in	d	InvS	2	out
And1	1	InvD	And	3	in	out	d	And1	2	in
And1	2	InvS	And	3	in	in	d	And2	2	in
And1	2	And2	And	3	in	in	e	And1	3	out
And1	3	Nor1	And	3	out	in	e	Nor1	1	in
And2	1	D	And	3	in	inout	f	And2	3	out
And2	1	InvD	And	3	in	out	f	Nor2	2	in
And2	2	InvS	And	3	in	out	g	Nor2	3	out
And2	2	And1	And	3	in	in	g	Nor1	2	in
And2	3	Nor2	And	3	out	in	g	nQ	1	inout
Nor1	1	And1	Nor	3	in	out	h	Nor1	3	out
Nor1	2	Nor2	Nor	3	in	inout	h	Nor2	1	in
Nor1	2	nQ	Nor	3	in	inout	h	Q	3	inout
Nor1	3	Nor2	Nor	3	out	in				
Nor1	3	Q	Nor	3	out	inout				
Nor2	1	Nor1	Nor	3	in	out				
Nor2	1	Q	Nor	3	in	inout				
Nor2	2	And2	Nor	3	in	out				
Nor2	3	Nor1	Nor	3	out	in				
Nor2	3	nQ	Nor	3	out	inout				
Q	1	Nor1	Cntct	1	inout	out				
Q	1	Nor2	Cntct	1	inout	in				
nQ	1	Nor2	Cntct	1	inout	out				
nQ	1	Nor1	Cntct	1	inout	in				

Figure 5. Component View and Net View.

Id, Nor1	Type, Nor	P 3	Pin, 1 2 3	N, 1 2 2	IO, in in out	IdC, And1 Nor2 Nor2 Q	IOC; out out inout in inout
-------------	--------------	--------	---------------------	-------------------	------------------------	-----------------------------------	--

Figure 6. Reduced datastructure for a View Object.

The view-object generator needs information other than the data, i.e., knowledge to perform its functions:

(a1) The query portion identifies the base data.

(a2) The specification of the primary relation identifies the object-entities.

(a3) The object prototype specifies the structure and linkages of the data elements within an object, and the access functions for the object.

Initially the view-object generators will be implemented as code, closely following the translation programs in use now to convert persistent storage representations of engineering data files into representations suitable for specific tools. For experiments a relational database can be used for storage of the base data, but expect that ongoing developments in EIS will provide systems with more appropriate functionality and performance.

```

enum io {IN, OUT, INOUT};
class Cpin;
class CCpin;
class ComponentsView
{
    char Id [8];
    char Type [6];
    short P; // Component pin count
    Cpin* Pin;
};
class Cpin
{
    io IO;
    short Q; // Connected component count
    CCpin* C;
};
class CCpin
{
    char* IdC;
    io IOC;
};

```

Figure 7. C++ datastructure for Component Objects.

University of Oxford Professorship of Computing Science

The University intends to make an election to the newly established Professorship of Computing Science. The Chair will be held in the Computing Laboratory and will be associated with the new joint honour school of Engineering and Computer Science. The new professor will be expected to enlarge the excellence and range of research in Computing Science at Oxford and will work in collaboration with Professors C.A.R. Hoare, J.M. Brady and K.W. Morton. The stipend of the professorship is at present £21,875 (under review).

Applications, naming three referees, should be received no later than January 19, 1987 by the Registrar, University Offices, Wellington Square, Oxford, OX1 2JD, U.K., from whom further particulars may be obtained.

Late Magazines? No Magazines? Membership Status Problems? No Answers To Your Complaints?

Let your
Computer
Society
Ombudsman
cut
through
the red
tape
for you.

Computer Society
Ombudsman
IEEE Computer Society
10662 Los Vaqueros Circle
Los Alamitos, CA 90720



The goal is to eventually provide non-procedural access for object-oriented approaches as well. By formalizing the semantics of the objects required by the tools, and interpreting the object type description, I believe that the view-object generators may be automated. The programmer then only provides the object type declaration and the Where clause defining the desired set of object instances.

With increased storage of data semantics in extended schemas one may advance further. With structural or dependency information about the base relations, automatic generation of the internal structure of an object type may become feasible. Since access to the objects by the tools is still procedural, the performance benefits of automated object generation would be minor. The major benefit is in the control of access and consistency that may be gained.

Justification

I am proposing a major extension to database and object-oriented concepts, with the intent to obtain the joint benefits that each approach provides separately today. The effort must be justified by these benefits.

Sharable access to objects. The proposed architecture supports procedural access to objects, as expected by object-oriented programming systems. Access to the base data is automatic, and nonprocedural. This permits base data to be effectively shared, since no single application or combination of applications imposes a structure on the base relations.

Growth of the system. New object types can be defined by adding new view-object generators. As is expected in databases, new data instances, types, and entire relations can be added without affecting other users' programs. Data can be reorganized without changing the object generator code since only the views will be involved. Such flexibility is essential for growth and multiuser access.

Support for a wide variety of representations. When simple tabular results are to be obtained from the database, a view-object generator can easily generate tables for direct inspection and manipulation. A graphics object-generator can project the attributes needed for visual display.

High-performance access from the database. To achieve this goal new database interfaces must be developed that make the benefits of the set-oriented database retrieval concepts available to programming languages. Current database interfaces create a

bottleneck when delivering data to programs. A common method for relational systems is to accept a query that specifies a set, but then require repeated invocations that deliver only single values or records at a time. This access mode, because of conventional programming techniques, is clearly inadequate.

The data from the databases is to be inserted into sets of objects at a time. For the object-oriented programmer such a set is best defined as a super-object. A program typically cannot proceed until the set is complete. The object generators need only be invoked once for all the data needed to compose a super-object. An effective view-object generator hence needs an interface at a deeper level into the existing relational functions.

Updating from objects. The programs can freely update the contents of the objects. Some applications require that these changes be made persistent and hence be moved from its object representation to the base database.

To achieve update I envisage a third architectural component, the view-update generator. This component can be invoked at commit-time, when results affecting the objects are to be made persistent. I envisage that such a process will only update from objects that have changed, and only replace values that were changed.

Where views have been constructed using joins, aggregation operators, and the like, automatic view update can be ambiguous. Restrictions on object update are one solution. However, as shown by Keller,⁴ such ambiguities can be enumerated and a choice can be made when the view-update is generated. The view update generator can also take advantage of the semantic knowledge available to an advanced view-object generator. An important source of knowledge constraining updates is authorization information.

Cost trade-offs

What costs and benefits will this architecture provide other than what I see as its structural advantage? I will now review areas where performance may be gained versus one of the two underlying approaches—database use or object-oriented programming—alone.

Set-based data access. A single invocation of a view-object generator will instantiate all specified objects. The overhead of programmed access to relations, typically requiring an initial Call giving the query and then iterated Calls to obtain the result piece by piece, is avoided. Also, no sequence of object instance-generating Calls, as seen in object-oriented program-

ming, is needed. Of course, one invocation of the object generator will be a major operation, but only one call should be needed per object type.

The object generator may also perform the so-called macro-generator function, where instances of design objects are generated based on a general prototype and parameters. The source of such information is a library of cell descriptions, permitting, say, the generation of a series of cells making up a register.

The view-object generator will be more complex than either a database retrieval alone or an object prototype. It will provide a very clear definition of the mapping from base data to objects and do so in a localized manner. A partial example was given, using the SQL approach to describe the view in Figure 5 and then using C++ to define the storage structure for the objects in Figure 7.

Binding of retrieved data. The ability of the view-object generator to bind the object will pay off in processing time. No joins are needed at execution to bind relational tuples, and no search operations are needed to assemble tuples belonging to the same object. Since the required information exists at view generation time, no search cost is expended. The only requirement is that the object's internal data structures permit retention of the information.

Task allocation that matches hardware system components. Implicit, but not essential, to my proposal are the complementary concepts of database servers and design workstations. They will be linked by a powerful, but often still, bandwidth-limited communication network. In such an architecture I expect that most of the retrieval and restriction operations will be carried out on the machine serving the database and the object generation and use will occur in the workstation.

Optimization in the file server. The file server can select optimum retrieval strategies and reduce the data volume needed to convey the information. Keeping data in sorted order and removing redundant fields as shown in Figure 6 is straightforward and effective. All operations are specified nonprocedurally and can be arranged and interleaved as needed to handle requests from the users. Concurrency control and version management are tasks best handled in the file server.

Communication minimization. The data volume is reduced in the file server. Communication bandwidth between server and workstation can be used fully for information, rather than for data to be ignored in the workstation.

High performance on workstations. The workstation only needs to assemble the information obtained into the desired object configuration. It is desirable that all objects for an application task can be placed into the real memory of the working processor.⁸ The objects now do not contain data fields irrelevant to the process at hand. Since these working objects are now compact, the probability that they will all fit into a modern workstation is increased. Virtual memory management can be invoked if needed, but the capacities of modern machines are such that there should be adequate memory space for working storage of the required objects. However, I can envisage several techniques to exploit having the data independence provided by view generators to improve locality for virtual memory management.

Physical organization of the database

Performance issues are considered critical in EISs, and experiments with relational databases have not given me confidence that adequate performance is easy to obtain. Performance will furthermore be impacted by the extensions, such as version support, that are needed in the engineering environment.³ Many of these extensions will also be useful for broader

objectives, so that there is a motivation to share the development and maintenance costs of relational database systems extensions.

Objects are seen as a means to solve the performance problem, because data is bound in a user-sensible manner. I believe that such binding can indeed be significant in workstations, although my own experiments have not given proof of that assumption when external storage is used.⁹ There are obviously many more factors to be considered simultaneously when building comprehensive systems, but storing data in relational tables is often seen as a critical issue.

The fully normalized model of the representation has as its objective the minimization of redundancy and the avoidance of a number of anomalies that can occur. It also supports a very simple storage structure, as seen in our example, that can lead to a large number of relations. Retrieval from such an organization will require a large number of joins, as seen in the example leading to Figure 5.

Updates seem to require less work in a relational database than in our proposal. However, when the database has to obey interrelational constraints, expensive joins must also be executed when the data are updated. For instance, it is necessary to ensure that the components exist and that the connections are valid when a net con-



LPI-PL/I lets you move your PL/I applications to today's hottest new workstations without going through the agony of re-programming.

LPI

LPI-RPGII, LPI-COBOL, LPI-PL/I, LPI-BASIC, LPI-FORTRAN, LPI-PASCAL, LPI-C, LPI-DEBUG
LANGUAGE PROCESSORS, INC., 400-1 TOTTEN POND RD., WALTHAM, MA 02154 (617) 890-1155

nection is to be changed.

There is, of course, no requirement for the physical storage structure to mimic literally the logical relational structure. Specifically, information from multiple tuples may be stored in one record. Denormalization has been formally considered, but is not supported by current DBMSs. It is commonly employed in practical relational databases. Preserving correctness in a denormalized database often requires additional transformation and care: records may have repeated fields and at other times some fields may be replicated in multiple records. Sometimes here work is required as well: dependent tuples, as connection points for a component, will automatically be deleted if they were implemented as a repeating group. I expect to profit here from ongoing research on non-first-normal-form databases.

In my architecture, where I expect that the object generator is the primary means for accessing data, complexity issues discouraging use of a non-normalized storage structure are moot. The object generators can be adjusted, probably eventually automatically, to any storage structure chosen. It is likely that the most efficient storage structure will be similar to the dominant object structure.

With such an organization I have provided the two primary objectives favoring the concept of object-oriented programs for computer-aided design:

(1) The concept of a view-object provides the desired clean and compact user interface.

(2) The flexibility of the storage structure can provide the locality needed to achieve high performance in source data access.

I can now provide these objectives for multiple users, and share selected information in a controlled fashion. I gain greatly from the flexibility obtained by interposing the object generator operating on a persistent base storage structure. I am no longer bound to an optimal data storage structure bound to an optimal object format. It is likely that in any design project the data retrieval loads will change over time. The object types that dominate use during initial design phases are not likely to be the object types used during the maintenance phase of the designed devices. A physical storage reorganization is now possible, as long as the view-object generators are adjusted correspondingly.

Replication, the primary tool to improve performance, can be utilized as well. A high-demand subset of the database can be replicated and made available in a performance-effective manner to the object generators. Object update functions can use primary copy token concepts to update all copies consistently and synchronously.

I argue that direct storage of objects, to make them persistent, is not appropriate for large, multi-user engineering information systems. I propose storage using relational concepts, and an interface that generates objects. To test the validity of the concept, initial versions of the interface may be coded directly. In the longer range I look toward knowledge-driven transformations.

Generation of objects from base data brings the advantages of sharing persistent information to an object-oriented program. It also provides a better interface from programs to databases, recognizing that access by programs, large or small, will remain essential for data analysis and complex transactions.

The separation of storage and working representation will also simplify the development of new approaches to engineering design.¹⁰ The object generator can be viewed as a system component utilizing knowledge to select and aggregate data for the information objectives. □

Acknowledgments

This paper has benefited from interaction with a VHSIC program-sponsored project to define information models for engineering information systems, although it certainly does not reflect the consensus of the group. Comments by Tony Gadiet of the USAF, WPAFB, Arnold Goldstein of MCC, and Professor Alice Parker of USC were constructive and helped focus the issues. Earlier presentations of these concepts at Workshops for Integrated Design and Manufacturing Processes at the Naval Postgraduate School in Monterey, Calif., April 1986, and at the Object-Oriented Database Workshop¹ engendered useful feedback. David Beech, from Hewlett-Packard, visiting the Stanford Center for Integrated Systems, provided motivation to consider the object-oriented approach and gave constructive criticism to earlier versions of this paper. He also provided the C++ example for Figure 7. Support for this research was provided in part by DARPA contract N39-84-C-0211 for Knowledge Based Management Systems and in part by the Air Force through a contract with UES F33615-82-C-1716.

References

I can cite here only a few of the many publications that have provided concepts leading to this paper. I include some recent global references to provide access to other work. Further references can be obtained from me.

1. K. Dittrich and U. Dayal (eds.), *Proc. 1986 Int'l Workshop on Object-Oriented Database Systems*, IEEE-CS order no. 734, Sept. 1986.

2. F. Lochovsky (ed.), *Object-oriented Systems*, special issue of the *IEEE-CS Database Eng. Bull.*, Dec. 1985; reprinted in *Database Engineering 1985*, IEEE-CS order no. 758, 1986.
3. R. H. Katz and T. J. Lehman, "Database Support for Versions and Alternatives for Large Files," *IEEE TSE*, Vol. SE10, No. 2, Mar. 1984, pp. 191-200.
4. A. M. Keller, "Choosing a View Update Translator by Dialog at View Definition Time," *Computer*, Jan. 1986.
5. M. Stonebraker and L. Rowe, *The Design of POSTGRES*, tech. report UC Berkeley, Nov. 1985.
6. B. Stroustrup, *The C++ Programming Language*, Addison-Wesley, Reading Mass., 1986.
7. H. M. Carter, "Computer-Aided Design of Integrated Circuits," *Computer*, Vol. 19, No. 4, Apr. 1986, pp. 19-36.
8. L. Holland et al., "Engineering Support System Software," *Micro*, Vol. 5, No. 5, Oct. 1985, pp. 17-21.
9. M. W. Wilkins et al., "Relational and Entity-Relationship Model Databases and Specialized Design Files in VLSI Design," *Proc. ACM-DA 22*, 1985, pp. 410-516.
10. D. J. Hartzband and F. J. Maryanski, "Enhancing Knowledge Representation in Engineering Databases," *Computer*, Vol. 18, No. 9, Sept. 1985, pp. 39-48.



Gio Wiederhold is an associate professor of medicine and computer science (research) at Stanford University. A member of the Computer Systems Laboratory and the Center for Integrated Systems, he is active in advanced database management and its applications to medicine and planning. On the editorial board of *IEEE Computer*, *ACM's Transactions on Databases*, *IEEE Expert*, and Springer-Verlag's *MD Computing*, Wiederhold has been involved in over 80 publications in computing and medicine.

Wiederhold received Lyceum and undergraduate training in aeronautical engineering in Holland. He held positions as programmer, manager, systems architect, and laboratory director. After 15 years in industry he returned to academia, obtaining in 1976 a PhD in medical information science from the University of California.

Readers may write to the author at the Department of Computer Science, Stanford University, Stanford, CA 94305.

Design and Performance Considerations for an Optical Disk-Based, Multimedia Object Server



Stavros Christodoulakis, University of Waterloo
Christos Faloutsos, University of Maryland

Optical disk-based, multimedia object servers will enable users to search gigabytes of information consisting of attributes, text, images, and voice.

This article describes work undertaken to automate storage and retrieval of complex data objects that contain text, images, voice, and programs. Until recently, the extremely large storage requirements of these objects posed a major problem in developing commercially viable database management systems capable of handling them. Database management systems that will manage multimedia information also have different functionality, interface, and performance requirements from traditional database management systems. Recent developments in hardware are now making automatic storage, retrieval, and manipulation of complex data objects both possible and economically feasible.

Present storage problems and suggested remedies

Optical disk technology. The use of optical disks has emerged as a revolutionary solution to the mass storage and retrieval of archival data. The main advantages are

- the very low cost per bit of stored information (four to five orders of magnitude less than the cost of storage on magnetic disks),
- long archival life (more than 10 years, versus the two to three years of storage provided by magnetic disks),
- very high storage density (one order

of magnitude higher than that provided by magnetic disks), and

- direct access capability.

Optical disks recently became commercially available. Typical optical disk products currently available on the market are capable of storage on the order of one gigabyte of data on one surface of a disk platter.

Large-capacity "jukeboxes" containing 64 two-sided optical disks are also commercially available. Jukebox prototypes¹ with very high transfer rates have been demonstrated for advanced scientific applications required by NASA.

Most commercially available optical disks are of two kinds: read-only or write-once. Some prototypes in advanced experimental stages provide read-write capabilities.

Several authors²⁻⁵ project that write-once optical disks will compete favorably with magnetic storage media for application environments that require storage of such unformatted data types as text, bit-maps, graphics, voice, and programs. Examples of such applications are

- office information systems,
- medical information systems,
- engineering information systems (including CAD/CAM),
- library information systems,
- geographic databases,
- consumer catalogues, and
- software reusability systems.⁶

Each of these application environments has a large archival component that makes

the write-once property of optical disks highly desirable.

We believe that in the future, very large repositories of information will exist within computer systems, and it appears to us that this information will be stored on optical disks.

Speech communication. A principal problem with such large repositories of unformatted information is the speed of man-machine communication.

We feel that voice storage—and possibly—limited voice recognition will greatly increase the man-machine communication bandwidth. They will make it possible for the user to insert information and, it is hoped, to absorb information faster. For example, many doctors would like to be able to dictate their observations about their patients in voice form, then store those observations in the system with such images as X rays and graphics. Users may be able to absorb information that is described in images and voice faster than information that is described in text.

Voice will become a very important means of man-machine communication.^{7,8} Voice digitization and products with limited voice recognition capability exist in the market today for reasonable prices.

Optical disks, because of their large storage capacity, are well suited for storage of both digitized voices and digitized images, which means that they provide an important means of integrating stored material.

Advanced workstation architectures.

Some other advances in hardware technology and changes in performance/cost trade-offs facilitate the development and use of large databases with unformatted data. These advances also indicate the trend towards use of digital systems. Workstations with powerful processors and with screens that are capable of displaying images of high quality (in various shades of gray and in color) are becoming affordable. An example is the Sun 3 family of workstations. They are capable of displaying on the screen pages of documents as these documents would appear on a high-quality output device, such as a laser printer.

Integrated communication services.

High-capacity communication networks make possible connections between workstations in a broadcast type of environment (such as Ethernet) or in a star type of environment (such as PBX). Some systems (such as Meridian) allow transfer of various types of digitized data, such as voice, images, text, and attributes.⁹ The trend in communication services, as in storage devices, is towards integrated digital systems.

Optical fiber communication networks are also appearing. They will alleviate problems of contention that arise from the high demands imposed by digitized data types in local area networks.

Emerging applications. With the advent of the new technologies described above, some high-technology centers in the US and in Japan are developing optical disk-based image-management prototypes. So far, the functionality of these prototypes is very limited. Integrated management systems for unformatted data are not available in the market.

Application environments that may require the storage and manipulation of unformatted data are

- office information systems,
- geographic databases,
- CAD/CAM and technical documentation systems,
- software reusability systems,
- some military applications,
- medical information systems,
- law information systems,
- tourist information systems,¹⁰
- advertising catalogues,
- newspaper- and magazine-production systems,
- large dictionaries and encyclopedias,
- systems used in education, and
- library information systems.

Research problems. The design and implementation of systems that manage unformatted data presents several challenges:

- First, good functionality and user interfaces should be provided.
- Second, the performance of these systems under a reasonable user load must be satisfactory.

We believe that in the near future, large information banks will exist, possibly utilizing optical disk technology. Two very important problems will be how to find the desired information and how to view it effectively. In an environment involving unformatted data, it is not easy for the user to specify precisely what he or she wants to see or not see.

The process of identifying the relevant information in such an environment is slow and difficult. Very powerful presentation and browsing facilities are required to increase the communication bandwidth between user and machine. The system's *data presentation manager* will be an important component in meeting this objective of communication between user and machine.

Several performance problems arise in an unformatted data environment when optical disks are used because optical disks have slower access times and higher densities of registration than magnetic disks.

Other sources of difficulty are the large volumes of data, the nature and complex interrelationships of the data stored, and the possible inability of users to specify very tight filters. (Filters allow users to choose objects for viewing at their workstations. See the section on "Overall architecture of the archival component," below.)

Architectural issues

We feel that future multimedia information systems will provide facilities appropriate not only for concurrent storage and retrieval of data (as database management systems do), but also facilities for creating multimedia information interactively, extracting information from large repositories of information, comparing extracted information, transforming extracted information from one form of presentation to another, and synthesizing new information from extracted and interactively created information.¹¹

In addition, these systems will provide facilities for viewing the information resource (that is, the database) of an organization as it existed at various times in the past. This would require that information not be destroyed by future manipulations. Information may eventually be retired from the on-line system.

Multimedia objects in a multimedia data environment may be in an *editing state* or in an *archived state*. Objects in the editing state may be modified. Multimedia objects in an editing state may be stored on magnetic disks (or possibly on rewritable optical disks in the future) that are shared among user workstations.

Retrieval is done by object name. Each user edits only a small number of these objects at any one time, and he or she can easily recall their names.

Objects in the archived state cannot be modified. If at some point a user feels that an object needs modifying, the user first extracts the object from the editing subsystem. The object, once modified, may be inserted in the archiver as a new object or as a version of the old object. Versions of the same object are associated by means of an underlining mechanism so that users can navigate from one version to another.

Overall architecture of the archival component.

We envision the overall hardware system architecture as being composed of (a) a multimedia object server subsystem and (b) a large number of user workstations interconnected through high-capacity communication links. The server subsystem may be composed of either several high-power processors, or multiprocessors sharing common mem-

ory. Archived multimedia objects are stored on write-once optical disks. The server subsystem may also contain some magnetic disks.

The software of the archival subsystem provides access methods, scheduling of concurrent retrieval requests and insertions, buffer management, data distribution management, version control, support for object "migration" and retirement, as well as other necessary database management functions, such as schema management, authentication, file management, and so on.

Users in this environment submit queries about object content from their workstations. The *query specification interface* allows a user to specify interactively certain information about the contents of objects so as to filter out objects that the user does not want to see. Frequently, these filters will contain either some text pattern that should appear within the text part of the object, or a part of a voice segment that can be recognized by a voice-recognition device. Sometimes users will not be able to specify very precise filters. (It is, for example, difficult for a user to specify the content of images.)

User queries are evaluated against the multimedia database by the server. The access methods supported by the server depend heavily on text retrieval methods.

A sequential retrieval browsing interface may be used to allow users to browse through *qualifying objects*. (Qualifying objects are those selected by the filter.) In this interface, *miniatures* of qualifying objects pass sequentially across the screen of the workstation. Miniatures are representations of the information in an object; they are meant to help the user to decide if the object is relevant. (They can, for example, contain (a) a small bitmap of the first visual page or (b) an indication that an object is an audio-mode object and some voice segments that are played as the miniature passes through the screen.) Several miniatures are projected at a time so that the user can concentrate more on interesting miniatures and skip fast through uninteresting ones.

When the user selects the miniature of an object, the *multimedia object presentation manager* undertakes the responsibility of presenting the information of the selected object. The manager's main objective is to help a user to browse effectively within a particular object. It is a software component of the archival subsystem; it resides in the workstations and requests the appropriate pieces of information from the data manager of the multimedia object server subsystem. The multimedia object presentation manager also helps the user navigate from the current object to other related objects by selecting appropriate links. The user may

interrupt this process and return to the sequential browsing interface or to the query specification interface to refine his or her filter.

Optimizing storage and retrieval from optical disks. The optical disk subsystem is an important component of the overall architecture. Its storage and retrieval performance affect user response time significantly. Methods for improving the storage and retrieval of multimedia data from optical disks should be developed and incorporated into the access component of the server.

Optical disks currently have slower access times than magnetic disks. An important performance difference between magnetic and optical disks is that optical disks can access *very fast data* (that is, data located near the current position of the disk head). This is done by moving a low-inertia mirror so that it focuses on a track near the data. The set of tracks that can be read without moving the access mechanism from its current position is called a *span*. A model for such a device and an optimal algorithm for retrieving objects from it has been described by Christodoulakis.¹² Exact and approximate estimates of retrieval performance for variable-length objects have also been derived.¹²

Simulation results show that clustering of objects on the disk surface may considerably improve performance when the optical disk access mechanism supports a span access capability. The improvement is larger for

- larger selectivities (that is, when there is a large proportion of qualifying objects) and
- larger span sizes (that is, when there is a large number of tracks per span).

A related problem is where to place a new object when the space that was originally allocated to its file is exhausted. Simulation results on the performance of some algorithms for placement have been investigated by Christodoulakis.¹³

Finally, object migration, data sharing, and version support are important aspects of multimedia object management systems. Support for these functions presents some differences from the traditional database environments because of the write-once property of optical disks and the possible large length of the objects, as well as the large variance in their lengths.¹³ The support for these functions in MINOS, a prototype multimedia object management system under development at the University of Waterloo, is described by Christodoulakis et al.¹¹

Contention for retrieval from optical disks may result in increased queueing delays and, therefore, longer response

times. The effect of contention may be reduced by appropriate scheduling algorithms. We are currently investigating several such algorithms in a distributed MINOS testbed.¹⁴

To be successful, multimedia object management systems must provide very good response times to users. To help achieve this, more research is required on the performance of an optical disk-based multimedia object archiver.

Symmetric multimedia object presentation and browsing

The multimedia object presentation manager is a very important component of the overall architecture. This component may reside in workstations, and it is responsible for effective presentation of and browsing within and among multimedia objects.

The facilities described in this section closely resemble those provided by MINOS.^{8,11}

Object-oriented architecture. Object-oriented architectures are appropriate for several application environments that make use of multimedia data. These architectures emphasize objects as the unit of access and manipulation. Methods that are applied to objects are well defined (that is, the methods evidence a strongly typed approach). A typical object-oriented interface allows the user to point to an object instance to find out the methods (procedures) applicable to the object.¹¹ Object-oriented approaches provide mechanisms to uniformly define, create, and relate objects and object interactions.

Objects can be associated with classes of objects, and classes can be organized into class hierarchies. Typically, the class concept is used to relate objects with common characteristics (*typing*), as well as for indicating the classes that an object is a member of (this indicating of classes is called *instantiation*). The organization of classes into class hierarchies allows properties of classes in the higher levels of the hierarchy to be shared by their descendant classes (a carryover called *property inheritance*).

In addition, an object-oriented architecture can provide mechanisms for describing the complex components of individual objects (the complex components form an *aggregation hierarchy*) and the interrelationships of individual objects. It can also provide mechanisms for keeping adjacent in storage (a) the components of complex and long objects and (b) the interrelationships of objects with other objects.

An object-oriented architecture frequently allows for independently active

objects. These objects are able to initiate some work by themselves. In its simplest form, an object-oriented architecture allows a user to communicate with an object to execute some procedure. (One example is a simulation in which the user inputs the parameters interactively and observes the results on the screen.)

To summarize, major characteristics of the object-oriented approaches are

- information hiding,
- abstraction,
- strong typing, and
- independently active objects.

It is also important to point out that major advantages of object-oriented approaches are incremental system development and easier replacement of the components of the system.

The unit of information in a multimedia data environment based on an object-oriented architecture is a *multimedia object*. Multimedia objects may be composed of *attributes*, an *object text part* (that is, a collection of text segments) an *object voice part* (a collection of voice segments), an *object image part* (a collection of images), and *object procedures*. Each one of these objects may be further subdivided into other objects. A unique *object identifier* is associated with each multimedia object. Multimedia objects may be related to other multimedia objects. Information about related objects is kept within each object itself.

Symmetric multimedia objects. Text and voice may be used interchangeably in multimedia objects to describe the same information. Moreover, they are both one-dimensional in nature. It is desirable to provide the user, who works interactively with the system, with symmetric capabilities for browsing through information described with text or with voice.

(Images, on the other hand, are two-dimensional in nature. A different set of browsing primitives is required.)

Browsing facilities for text and voice may utilize the *presentation form* of text or voice, the *logical components* of text or voice, *links* between objects, or *pattern-matching* primitives.

In speech, differences in emphasis and meaning are expressed by a speaker by employing various methods, such as increased loudness, different intonations, variations in the length of pauses between words, and so on. In text, the same emphasis and meaning aspects are expressed by some special symbols (for example, the exclamation point, question mark, comma, or period) as well as by some conventions, such as underlined words, italicized words, boldface words, and so on.

Text presentation must support such symbols and conventions.

Browsing by means of the presentation form. The presentation form of text is subdivided into text pages. A *text page* is all the text information that is presented at the same time on the screen of the workstation. Often text is intermixed with images on the same page. We call these pages, which may contain both text and images, *visual pages*. When a user browses through text (in a paper document or at the workstation) he or she typically has an idea of how far within the text to look next, and based on that the user estimates how many pages to advance. Figures 1 and 2 show examples of visual pages found in objects managed by MINOS.

The page concept is required for voice data to achieve the same effect. *Audio pages* (or *voice pages*) of a speech are consecutive *partitions* of the audio part of an object and are of an approximately constant time length. The user can advance several voice pages at a time to find relevant information.

A difference between speech and text that may be incorporated into a system is not interrupting speech at the end of each voice page. In contrast, visual pages are not turned automatically, but only by explicit selection of a menu option. The reason for explicit selection is that the time required for reading a text page is different from user to user.

Visual-page browsing capabilities allow the user to move to the next page, the previous page, to advance a number of pages back and forth, and to find a page with a given page number.

Voice browsing capabilities allow the user to interrupt the voice output, resume the voice output from the current position, resume the voice output from the beginning of the current voice page, as well as to browse between pages in a fashion similar to text browsing (that is, with facilities for selecting the next page, the previous page, and so on).

Browsing by means of the logical components. A difference between reading text and hearing voice is that text pages present a "cache" of information to the user; that is, the user can read twice the same word, sentence, or paragraph if he or she did not understand its meaning. The user can even look at a previous paragraph in the same page to understand better what he or she is reading.

A similar facility in voice would allow the user to interrupt the speech, and then use instructions such as "select previous word," (or previous sentence, previous paragraph, and so on).

It is, however, very difficult to provide a voice browsing facility that will work automatically. Detecting word boundaries, the ends of sentences, and ends of paragraphs automatically from digitized-voice data

when an unlimited vocabulary is allowed is very difficult (at least with current and near-future voice-recognition and AI technology). Moreover, the user may not be absolutely sure about paragraph or sentence boundaries in uninterrupted speech when he or she inputs voice material for browsing.

Pause is a segment of digitized voice that does not contain any sound. (In practice, the intensity of the digitized voice itself is very small.) Short and long pauses may be distinguished from digitized voice. When the user browses through voice segments, he or she may specify that the audio be replayed, starting some number of short or long pauses back from the current position.

The length of the short pause may roughly correspond to the average length of a pause between word boundaries, while the length of the long pause may roughly correspond to the length of a pause between paragraphs. The exact timing for short and long pauses depends on the speaker and the particular section of the speech. The speaker decides the timing by sampling the current speech context. Of course, there is no guarantee that the short and long pauses, as input, will match word boundaries and paragraph boundaries, respectively.

The user may, however, recall recent pauses. The combination of the user's memory and the audio replay capability provides an always available form of browsing that can be used in current and adjacent speech areas and is independent of any manual editing done as part of voice input.

A text segment of a multimedia object may be logically subdivided into *title*, *abstract*, *chapters*, and *references*. Each chapter is subdivided into *sections*, sections are subdivided into *paragraphs*, paragraphs into *sentences*, and sentences into *words*. For objects that have been generated interactively in a given environment, these subdivisions can be easily identified by the tags that the user inserts to format the text.

A voice segment of a multimedia object may also be subdivided into logical components, as is the case with text. Browsing capabilities in text or in voice allow the user to see or hear the page that begins with the next logical unit or a previous logical unit (such as a chapter, a section, and so on).

The logical components of voice may be manually identified at the time of the insertion by pressing the appropriate buttons, or they may be identified later. When the logical components are identified at insertion time, the speed with which the user can insert information in the computer is reduced, and, in addition, the user's hands become occupied with pressing buttons.

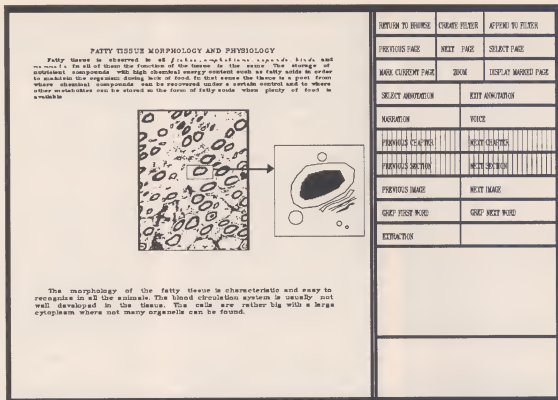


Figure 1. A visual page displaying browsing within an object interface of MINOS. The page combines information forms: text, bitmap, and graphics.

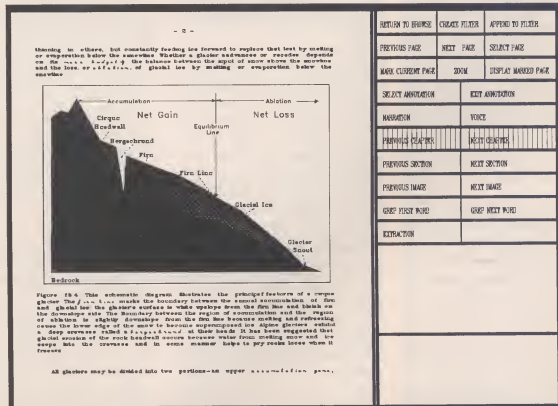


Figure 2. An example of "logical text" in MINOS. Such text is all the information that is logically related to an image or other presentation form. As much of it as will fit is displayed on the same page under the image when the "next page" or some other such button is selected.

There are reasons other than occupying the user with pressing buttons that may make it undesirable to edit manually all incoming information. Some information is clearly more important than other information, and it may deserve to be edited if this will help other users when they browse through it. The degree of editing should vary according to the importance of information. For example, in a certain object, only identification of chapters may be desirable. In another, identification of chapters, sections, and paragraphs may be desirable. A similar situation arises as to the degree of initial editing necessary for text information when this information has been inserted in an object as a collection of bitmaps of pages by means of an image-capturing capability.

The logical browsing options available to the user may depend on the object (for example, on the logical units that have been identified for the object). The menu options that are displayed define the set of available operations (for example, next chapter and so on).

Browsing by means of pattern matching. The third type of browsing through text and voice information is based on pattern matching. A user types a text pattern or speaks a voice pattern that is recognized, and the system turns to the first page with the occurrence of this pattern in the object's text or voice. Voice recognition may not take place at the time of browsing. Instead, some voice segments may be recognized from the digitized voice at the time of voice insertion, or during the machine's idle time. The recognized voice segments are used to provide content addressability and browsing capabilities that incorporate the same access methods used for text. Here, as with the editing of text, the extent to which this capability is used depends on the importance of the particular object. Two aspects are of interest:

- First, even limited voice recognition can be used to reduce or eliminate the need for manual indexing, which would be necessary for both retrieving objects by means of their content and browsing within a particular object.
- Second, recognized utterances can be associated with a particular point of the object's voice part to facilitate browsing within an object.

Image-driven objects versus audio-driven objects. Each multimedia object has a *driving mode* associated with it. The driving mode is the principal way of presenting the information in the object, and it can be either *visual* or *audio*. Visual-mode objects have a principal presentation form that is based on visual pages. The "next page" command for visual-

mode objects always implies the next page of visual information, whether of text or images. The same is true with all the other page-browsing commands.

Audio-mode objects have a principal presentation form that is based on audio pages. The "next page" in these objects refers to the next audio page.

A driving mode is enforced for each multimedia object so that users do not become confused trying to navigate in two different media at the same time.

Voice logical messages are unstructured, typically short, audio segments. They can be attached to either visual-mode objects or audio-mode objects. When attached to visual-mode objects, they may be associated with either text segments or images. (Text is linear. Two points identify the beginning and the end, respectively, of a text segment to which a voice logical message is attached. The two points may coincide.) When attached to audio-mode objects, voice logical messages may be associated with voice segments or with particular points within the object's voice part. The semantics dictate that the voice logical message be played when the user first branches into the message's corresponding text or audio segments during browsing. In the case of audio-mode objects, the voice logical message is played before the voice of the related segment. Voice logical messages may be attached to overlapping text segments or overlapping images.

Audio-mode objects can be used with visual logical messages to describe the same information that can be described by objects composed of text and images. Consider, for example, a doctor who wants to file his observations about an X ray of a patient. He or she may find it easier to insert this information as an audio-mode object. (Doctors are notoriously bad typists!) The X ray can be attached as a logical visual message to that section of a *voice object* which relates to the X ray. (A voice object is an object that has voice components only—no visual components.) At presentation, the X ray will appear on the screen of the workstation only during the related section of the speech. In addition, if the user branches during browsing to some section of the speech that relates to the X ray, the X ray will automatically be displayed.

In the previous example, all the voice information that is related to a particular image may be played while the image is displayed on the screen of the workstation. This way the user can see the image while he or she hears the observations about the contents of the image.

Visual logical messages are short segments (at most one visual page long) of visual information (that is, of text and/or

images). They are unstructured in the sense that they are always displayed in the same part of the presentation form (that is, in the top part of the display area). They can be attached to either audio-mode objects or visual-mode objects, just as audio logical messages can.

When they are attached to audio-mode objects, the semantics dictate that the visual logical messages stay on display for the duration of the play of each voice segment to which they are attached.

When visual logical messages are attached to visual-mode objects, the semantics dictate that the logical message be displayed at the upper part of the screen while the lower part of the screen be devoted to the display of a part of the related visual segment. When the user turns a page, another part of the related segment is displayed in the lower part of the page while the visual logical message remains at the top of the page. The user has the option to specify that the visual logical message be displayed only once when, from anywhere within a nonrelated segment, the user branches off to a related segment.

A symmetric functionality can be achieved for a visual-mode object by attaching visual logical messages to sections of it. For example, the visual logical message, which is displayed by default on the upper part of the screen, could be an X ray. The lower part of the screen could be used for displaying the related text information—containing, say, the observations of the doctor. A user (presumably the doctor; at another time, other doctors might be users) could browse through the related text by keeping the X ray in front of him or her continuously. This way, the doctor would not have to turn the visual pages back and forth to associate the meaning expressed in the text with the contents of the image (which usually needs to be done with paper documents). The X-ray bitmap is stored only once within the multimedia object.

Object acquaintances. *Relevant objects* are objects containing information related to information in a section of a *parent* object. Relevant objects are independent multimedia objects (that is, they exist by themselves), in contrast to voice logical messages and visual logical messages, both of which exist only as part of a multimedia object. An object may have several relevant objects (including itself), each one having some information related to a part of the parent object. The user does not automatically see the relevant objects (in contrast to logical messages, which are displayed automatically).

A *relevant object indicator*, which is displayed on the screen of a workstation,

indicates the existence of a relevant object. The user can browse through a relevant object by explicitly selecting the relevant object indicator with a mouse. The driving mode of the relevant object may be different from the driving mode of the parent object. The user can browse through the information of the relevant object by using the driving mode of the relevant object. He or she can return to the parent object by explicitly selecting the *return from relevant object* indicator. At this point, the parent object's mode of browsing is re-established.

Users can browse through a relevant object by using the menu options for browsing through a multimedia object (as described) or by searching for *relevances*. Relevances are sections of text, sections of voice, and parts of images in the relevant object that are related to the content of a particular section of the parent object. Relevances to text sections of the parent object are indicated graphically with beginning and end indicators. Relevances to images are indicated by polygons displayed above the image. Relevances to voice segments are indicated by playing of the voice segment independently. (A menu option has to be selected if the user is to hear the next related voice segment.)

Relationships between objects (in terms of relevant objects and relevances) provide a powerful means of browsing through a single object or navigating between multimedia objects that contain some related information. One important facility allows the user to browse through a bit of related information that has been inserted into the computer system by means of various modes (for example, primarily by visual or primarily by audio). The presentation manager requires the explicit selection of a relevant object indicator to allow the user to browse through an object that may be in a mode different from the one in which the user is working. The user can return to the original object by explicitly selecting a *return-from-relevant-object* indicator. These selection requirements keep the user confident of where he or she is and what he or she is doing at every moment.

In many applications, relevances are useful for displaying information that is relevant to a particular section of the objects involved. Consider, for example, a set of images describing various levels of an engineering design. Parts of an *image object* (an object that consists wholly of images) that are drawn at a certain level of description may correspond to one or more objects at a different level of description. The user may want to identify the corresponding objects. This can be done easily by associating a relevant object indicator with the first object. When the indicator is selected, the related image is displayed and

a set of polygons projected on top of the screen. These polygons identify all the corresponding objects.

Transparencies. *Transparencies* are visual pages that allow the user to see the current visual page as well as the previous visual page displayed on the screen of the workstation. They function in a manner similar to the transparencies used by a speaker.

A *transparency set* is an ordered set of transparencies. The multimedia object designer may specify one of two ways for displaying the transparencies of a set.

The first method superimposes every transparency on the top of the previous one, and all of them on the top of the page previous to the transparency set.

The second method displays every transparency of the set separately, and each transparency is superimposed on top of the page previous to the transparency set.

The user may alter the presentation order specified by the document designer, and he or she may choose to see only certain transparencies of the set at the same time. The user can do this by displaying the transparencies independently (that is, by using the second method) and selecting transparencies that he or she wants to see superimposed over each other.

Transparencies are typically very useful for displaying information on top of images. They are also useful for displaying information (for example, an alternative set of values for some measurements from an experiment) on the top of text information. For MINOS, however, we could not find real-world analogs to suggest a similar mechanism for the top of voice pages.

Transparencies and transparency sets are useful in many application environments. As part of an office's filing system, transparencies may be used to create multimedia documents that simulate a speaker who superimposes transparencies over one another to interrelate information (for example, to compare values from an experiment by projecting different sets of results on the same axes). Voice logical messages may be associated with each transparency to further simulate a speaker. This use of transparencies and transparency sets is a much more effective method of presenting information than reading sequential text. The result may be increased man-machine communication bandwidth. Using transparencies and transparency sets in this way is also desirable for future, computer-resident textbooks.

In a medical information-system environment, a doctor can use this capability to file an observation made about an X ray. In an audio-mode object, the X ray (which is typically a large bitmap) may be a

visual logical message associated with a segment of voice. Transparencies that are also visual logical messages can be used to pinpoint particular areas within the X ray that relate to some subsection of the speech.

A symmetric capability can be achieved in visual-mode objects by superimposing (a) transparencies that contain a polygon or a circle (to identify an area of interest) on the X ray, and (b) some related observations in a text form under the image of the X ray.

Complex images. Images in a multimedia information system environment may be bitmaps or graphics. Images with graphics contain *graphics objects*, such as points, polygons, polylines, circles, and so on. Graphics objects may have a *label* associated with them. A label is some short piece of information about the object. The presentation form of a label may be *invisible*, *text label*, or *voice label*. A text label is a short piece of text that is associated with a graphics object, and a voice label is a short voice segment that is associated with a graphics object. Text labels are displayed near the graphics object, at a position specified by a designer. A *voice label indication* is displayed, in a position specified by the designer, near a graphics object that has a voice label. Invisible labels may be text labels or voice labels.

Voice labels are not played automatically. This does not make them incompatible with text labels: The text label may be displayed on an image, but the user may not read it (even if he or she browses through the image). The user reads textual information only when he or she focuses on it. He or she simulates this focusing of attention with regard to voice labels by using the mouse to select the appropriate voice indicator. The user may also request that all voice labels be played, in which case the system plays them in a system-defined order.

Labels may be used to identify corresponding objects within an image. The user can specify a pattern and request that the objects found within a certain label that conform to this pattern be highlighted. This facility is useful for browsing through large images with many objects, such as a road map. The inverse facility may also be provided. The user can then select an object with the mouse, and the system will play or display the label associated with the object.

Navigation within large images. Images are two-dimensional in nature. In very large images, the user may want to see a small portion of the image at a time. He or she can use a window to browse through the image. The system will only retrieve the relevant data.

A *view* is a rectangle overlaid on an image. The portion of the image that is enclosed by the rectangle is presented in such a way that it utilizes a major part of the screen's display area. The view can be moved to the top of the image by means of menu options and the mouse. If the voice option has been turned on, the system plays the voice labels that are encountered as the view moves. Noncontiguous moves (jumps) within the view can also be specified by choosing menu options. The dimensions of the view can be shrunk or expanded in small increments by selecting the appropriate menu options and defining the size of the new rectangle with respect to the old size. When the size increases, new labels may be played.

View definition can be done in the area above a *representation* of the image. A representation of an image is a high-level depiction in which the contents of the image are presented in positions that correspond to the actual positions of the objects of the image (in other words, it is a miniature). The representation of the image is much smaller than the image itself, and thus it is easily transferred to main memory and projected on the display. When a view is defined on the representation, the system has to transfer only the view data into main memory and not the whole image (the whole image must be transferred when a user retrieves all the data of the image and then zooms to the desired data). The system indicates explicitly that an image is a representation, and the menu options displayed when a representation is shown allow for the definition of a view and the retrieval of the related data from the image.

A *tour* is a sequence of views of an image; it is defined by the designer of the multimedia object. The sequence is played automatically; the user does not need to press the "next page" button. A tour is defined by (a) a rectangle and (b) a sequence of points indicating the position of the rectangle on the large image or on a representation of it. A logical message (visual or audio) may be associated with each position of the tour. The user may interrupt the tour and move the window around to navigate through other positions of the image.

An *overwrite* is a visual page with an image containing a number of bitmaps or graphics objects; these bitmaps and graphics objects may be shaded. When the overwrite page is turned, the bitmaps, lines, and shades of the overwrite image replace their counterparts on the previous page, but they leave anything else intact.

Views facilitate retrieval of pieces of information from very large images without retrieval of the whole image. This facility is useful for retrieving information from very large images containing engineering

designs (such as CAD/CAM images) or for retrieval of information from large bitmaps.

Tours facilitate the automatic movement of a window within a map. If a voice logical message is associated with each of the views, the overall effect is to simulate a guided tour through various sections of the map. This facility is useful, for example, in tourist information systems.

Associating labels with objects is useful for locating objects within a large image, as well as for enhancing content addressability and browsing capabilities. Such facilities may, for example, be found desirable in real estate databases, tourist information databases, city maps, and large engineering designs.

Process simulation. In *process simulation*, an ordered set of consecutive visual pages is automatically displayed one page after the other—that is, the user need not press the "next page" button. Logical messages may be attached to each page. When audio messages are attached, the next visual page is shown only after the audio logical message has been played. The relative speed by which pages are placed one on the top of another is set at object-creation time, but it may be altered by the user. The automatic page change capability may be used to give the impression of a contingent move or change, thus simulating a process.

Process simulation provides the means for multimedia object designers who are not programmers to simulate a process. This is done by having the system automatically turn pages at a specified speed. Each new page can be a new image, a transparency, or an overwrite. The new page may have a voice logical message associated with it.

Examples of applications are many. Process simulation provides an easy way to "program" some forms of animation that can be used easily by multimedia object designers who are not programmers. It could be used as a manual to explain an installation process. It could be used to explain a factory process to new employees or to visitors. It could be used to explain the evolution of a battle in tourist information systems or in textbooks. Finally, in a medical information system, it could be used to simulate the propagation of some medicine or disease through the human body.

Procedures. Procedures may be included as parts of complex multimedia objects. The user may interact with procedures in order to input certain parameters. The user may see the results of execution of procedures projected on the screen of the workstation. An example of an application where this facility is useful is a tech-

nical document that allows users to study the effect of various parameter values on an experiment.

Procedures may also be associated with the points composing an image object, which enables the user to decide how this object is to be displayed on screens with different resolutions.

Performance issues for content searching

Searching according to content is an important capability of a database management system. While working with a multimedia-object system, a user may partially remember (or be able to specify in part) the contents of an object he or she is interested in; the system should accept such a partial description and return the qualifying objects. The methods for accessing formatted data are well known.^{15,16} On the other hand, content addressability for unlimited recognition of voice and image data is a difficult problem.

In this section, we will concentrate on access methods for the text part of objects. (We anticipate that text will be a very important means of specifying the content of multimedia objects.) Many methods have been proposed in the literature. However, they seem to form the following large classes:

- **Full text scanning.** In this class, the user inputs a search pattern, and the whole database is scanned until the qualifying text parts are discovered and returned to the user. The method requires no space overhead, and minimal effort on insertions and updates. However, despite the existence of some fast string-searching algorithms,¹⁷ scanning of a large database may take too long.¹⁸

- **Inversion.** This method incorporates an index. An entry in this index consists of a word, stem, or concept with a list of pointers affixed. These pointers point to objects that contain the word, stem, or concept. Many commercial systems have adopted this approach: STAIRS,¹⁹ MEDLARS, ORBIT, and LEXIS,²⁰ to name a few. The main advantage of the method seems to be its retrieval speed. However, it may require large storage overhead for the index: 50 percent to 300 percent of the initial file size, according to Haskin.²¹ Moreover, insertions of new objects with text parts require expensive updates of the index.

- **Signature files.** In this class of access methods, the text parts of the objects are stored sequentially in the "text file."²² Their signatures are stored sequentially in the "signature file." The signature of an object contains condensed information about the object. The signatures are created

with hashing methods. (These methods are illustrated below.) When a query arrives, the system scans the signature file sequentially and the signature of each object is examined against the signature of the query. Thus, many nonqualifying objects are discarded. The rest are either checked so that the "false drops" will be discarded or they are returned to the user as they are. An object is called a "false drop" if it does not actually qualify in a query when its signature indicates that it should. This method is faster than scanning the full text, mainly because the size of the signature file is much smaller. However, it is slower for large files than the inversion method.²² The signature file method requires much less space overhead than inversion (approximately 10 percent, according to Christodoulakis and Faloutsos²³), and it can handle insertions easily. If carefully designed, the signature file method can handle queries based on parts of words and can tolerate errors.²⁴

- **Clustering.** In this method, similar documents are grouped together to form clusters. They are usually stored together physically, too. Clustering has been extensively examined in the literature of library science.^{25,26} The space overhead is rather small— $O(\log n)$, where n is the number of documents. Similarly, the retrieval time seems to be proportional to $\log n$. However, it seems that clustering cannot handle insertions easily, because insertions may necessitate the reorganization of the cluster structure.

- **Multi-attribute hashing.** Gustafson²⁷ has proposed a multi-attribute hashing scheme based on *superimposed coding*. The main characteristic of the scheme is that the search time decreases almost exponentially with the number of terms in a conjunctive (or AND) query. Although elegant, this method has practical limitations: The text part has to have a vocabulary of fixed size, performance deteriorates as files grow (as in every nonextendable hashing method), and queries other than conjunctive ones are handled with difficulty.

The office environment, in which we are primarily interested, has the following features:

- Large databases: Typically, these have a 1-gigabyte size, according to Bird, Tu, and Worthy²⁸, or a 65-gigabyte size, according to Hollaar et al.²⁹
- Large insertion rates, but few deletions and updates.²⁹
- Most of the objects in an office are seldom required after they have been filed. Graving³⁰ reports that the frequency of accessing an item decreases very fast with its age.

In the office environment, the signature file method seems to be a reasonable

Text part	data	base	management	system
Word signature	0000	0100	0111	1011
Object signature	0000 0100 0111 1011			

Figure 3. An illustration of the word signature (WS) method. The object contains four words only. Each word yields a 4-bit word signature ($f=4$).

Word	signature			
data	0010	0000	1100	0100
base	0000	0100	1010	0010
management	0101	0000	1100	0000
system	0001	0001	0110	0000
Object signature	0111	0101	1110	0110

Figure 4. An illustration of the superimposed coding method for accessing text. Words per object: $d=4$; signature size: $F=16$ bits; bits per word: $m=4$.

choice. By dividing the objects into mutually disjoint files (for example, "correspondence 1984," "internal memos 1985," and so on²³), one can create object files of manageable size in which the signature file method will provide fast access.

An important advantage of the signature file method over inversion is that the method can be used with data on optical disks, which are ideal for archiving.³ Write-once optical disks do not allow rewriting of a block. This is a problem for the inversion method because blocks of the index have to be rewritten with every insertion or batch of insertions. Signature files do not create any problems because the signatures of the new objects can be appended easily to the end of the signature file.

The other alternative, full text scanning, may be too slow because the method involves the retrieval of a large number of blocks. Signature files can provide an almost 10-fold speedup over full text scanning—with only approximately an extra 10 percent space overhead.³¹ The speedup is brought about by the reduced I/O requirements (the signature file is smaller than the text file by an order of magnitude) and the simpler operations required for searching the signatures.

Creating signatures. Now let us go into more detail on creating signatures. Signature extraction methods for text only have been examined before.³² The most remarkable methods are *word signatures* (WS), *superimposed coding* (SC), and methods based on *compression*. To illustrate these methods, we shall use objects with text parts containing $d=4$ words each.

The first method^{33,34} requires that each word of the text part of an object be hashed into a bit pattern of length f . These patterns ("word signatures") are concatenated to form the object signature (see Figure 3). Searching is performed in the obvious way. For example, in a single-word query, the signature of the search word is extracted and all the object signatures are searched. Those that contain the signature of the search word are retrieved. To improve the performance, common words (for example, "the," "a," and so on) may be ignored. In the rest of this article, we shall refer to this method as "WS."

Superimposed coding works as follows: Each word (string) yields a bit pattern of size F . These bit patterns are "OR-ed" together to form the object signature. The word signature creation is rather sophisticated and needs more detail: Each word yields m bit positions (not necessarily distinct) in the range 1- F . To create the word signature, the corresponding bits are set to "1," while all the other bits are set to "0." For example, in Figure 4, the word "data" sets to "1" the third, ninth, tenth, and fourteenth bits ($m=4$ bits per word).

When the system searches for a word, the signature of the word is created, setting to "1," for example, bits in positions 2, 3, 6, and 9. Each object signature is examined. If the particular bit positions (in this case, 2, 3, 6, and 9) of the object signature contain "1," then the object is retrieved. Otherwise, it is discarded. The signature size F directly affects the number of false drops, while the m selected must allow approximately half of the bits in an object signature to be "1."

data	0000	0000	0000	0010	0000
base	0000	0001	0000	0000	0000
management	0000	1000	0000	0000	0000
system	0000	0000	0000	0000	1000
Object signature (uncompressed)	0000	1001	0000	0010	1000

Figure 5. An illustration of the compression-based methods. Size of the bit vector: $B=20$; $m=1$ bit per word. The resulting bit vector is sparse and can be compressed.

Notice that SC is sensitive to variances in the number of distinct words d per object. For example, an object with a vocabulary of $d=10$ distinct words will result in a signature almost full of '1's.' A solution to this problem is to divide the text part of the object into "logical blocks,"²³ that is, into pieces of text containing a fixed number of distinct words (d in our case).

The next methods are based on compression. The idea is to use a large bit vector of B bits and to hash each word into one or more (as an example, let us say m) bit position(s), which are set to "1" (see Figure 5). The resulting bit vector will be sparse, and therefore it can be compressed. Methods for compressing sparse vectors are

- *Run length encoding (RL).*³⁵ In this method, the number of zeros between two successive "1's" in the sparse vector is encoded.

- *Bit-block compression (BC).*³² The idea with this method is that the sparse vector is divided into groups of consecutive bits (*bit-blocks*), and each bit-block is compressed individually. This method requires more space than RL, but it makes searching faster. The size b of the bit-blocks is fixed for all objects in the BC method.

An extension of the bit-block compression method is the variable bit-block compression method (VBC), which avoids splitting the objects into logical blocks. The idea is to allow different (that is, *variable*) bit-block sizes for each object, according to the size of the object's vocabulary.

Analysis of the above methods' performance³² shows that, for the purpose of decreasing false drop probability for a given signature size, the best method is RL, followed in order of descending value by BC, WS, and SC. False drop probability is the "error rate" of the signature file: the proportion of nonqualifying objects that pass the signature test.

From a practical point of view, there are some additional considerations. The most important of them are

- (1) the speed at which the different methods can search an object signature,
- (2) the performance of each method as regards the more complicated queries,
- (3) the ability of each method to answer queries based on parts of words, and
- (4) the ability of each method to preserve sequencing information.

Let us discuss briefly these four points:

- (1) The fastest method for searching a signature (given that it has been brought into main memory) seems to be SC: It requires only m (typically on the order of 10) bit comparisons to accept or reject a signature in a single-word query. The BC method requires some more bit comparisons (on the order of 40 total), while the RL method requires that approximately

half of the encoded zero-intervals be decoded and added, thus necessitating longer search time (for approximately 300 bit comparisons and for calculations for the decoding). The WS method requires that the whole block signature be examined (approximately 600 bit comparisons), but it does not need any decoding or any additions. These values correspond to objects with text parts spanning approximately one typed page; the estimated vocabulary is $d=40$, and the target false drop probability is 10^{-3} .

(2) All the signature methods do well on conjunctive (that is, AND) queries. Methods that split text into logical blocks (that is, SC, BC, and RL) require more bookkeeping than the rest of the methods (such as the WS method when employed without logical blocks, and the VBC method).

(3) At the present time, only SC²³ can handle queries on parts of words.

(4) Only the WS method preserves the sequencing information.

As a final conclusion, note that the VBC method seems to perform well if the text part of objects is of variable length and if queries refer to several words. Otherwise, the SC method seems preferable to it.

Notice that WS, SC, and the compression-based methods (VBC, BC, and RL) can be extended to handle attributes: The main idea is to use a different number of bits m for each attribute, according to the different attribute occurrence and query frequencies.^{36,37} Thus, attributes that appear often in queries (such as, say, "sender" or "author" and so on) should set more bits to "1" in the object signature.

In this article, we have described some issues related to the architecture of a multimedia object server. The server manages complex multimedia objects that may contain attributes, text, images, voice, and programs.

We have presented examples from MINOS, a system which is under development at the University of Waterloo. An integrated version of the MINOS system that manages multimedia documents is described by Christodoulakis et al.¹¹ Detailed design, implementation, and experimentation with more general object types and relationships between objects (such as those described in this article) is currently underway.

Multimedia objects in a multimedia object server environment may be in an editing state or in an archived state. Objects in the archived state are managed by the multimedia object server. The server we envision stores multimedia objects on write-once optical disks. It provides functions for querying, access methods based on object content, and powerful presenta-

INTERNATIONAL JOURNAL OF PARALLEL PROGRAMMING

(formerly the *International Journal of Computer and Information Sciences*)

Editor-in-Chief:
Gary Lindstrom

The *International Journal of Parallel Programming* (IJPP) covers new developments in parallelism from a programmer's perspective.

Sample areas addressed in IJPP include linguistic foundations, conceptual frameworks, high-level languages, evaluation methods, and implementation techniques.

Subscription: Volume 15, 1986

(6 issues) Institutional rate: \$195.00

in US/\$218.00 elsewhere

Personal rate: \$60.00 in US/\$69.00 elsewhere

Volume 16, 1987 (6 issues)

Institutional rate: \$215.00 in US/

\$240.00 elsewhere

Personal rate: \$60.00 in US/\$69.00 elsewhere

Plenum Publishing Corporation

233 Spring Street

New York, N.Y. 10013-1578

In the United Kingdom:

88/90 Middlesex Street

London E1 7EZ, England

Join the Computer Society Today

Schedule of Fees

Membership dues and publication subscriptions are annualized to December 31. Depending upon the date your application is received by the Computer Society, pay the full annual amount or one-half the annual amount as indicated below. Membership expires December 31.

	Half-Year Mar 1-Aug 31	Full-Year Sept 1-Feb 28
1 I don't belong to the IEEE and I want to join just the Computer Society	☐ \$19.50	☐ \$39.00
2 I already belong to the IEEE and I want to join the Computer Society IEEE Member Number _____	☐ \$ 7.50	☐ \$15.00
3 I don't belong to the IEEE and I want to join both the Computer Society and the IEEE* Full IEEE members receive <i>Spectrum</i> , <i>The Institute</i> , and many additional benefits. (Total amount to be paid includes annual dues and regional assessment, if any.)		
I reside in Region 1-6 (United States)	☐ \$41.00	☐ \$82.00
I reside in Region 7 (Canada)	☐ \$38.50	☐ \$77.00
I reside in Region 8 (Europe, Africa, or the Middle East)	☐ \$37.00	☐ \$74.00
I reside in Region 9 (Latin America)	☐ \$33.50	☐ \$67.00
I reside in Region 10 (Asia and Pacific)	☐ \$34.00	☐ \$68.00

*ACM members who join both IEEE and the Computer Society are entitled to a deduction of \$5 off the full-year rate, \$2.50 off the half-year rate

4 In addition to my automatic subscription to *COMPUTER*, I would also like to subscribe to:

(issues per year)

IEEE Expert (3151)	4	☐ \$ 6.00	☐ \$12.00
IEEE Computer Graphics and Applications (3061)	12	☐ \$13.00	☐ \$26.00
IEEE Design and Test (3111)	6	☐ \$ 8.50	☐ \$17.00
IEEE Micro (3071)	6	☐ \$ 8.50	☐ \$17.00
IEEE Software (3121)	6	☐ \$ 8.00	☐ \$16.00
Transactions on Computers (1161)	12	☐ \$ 9.00	☐ \$18.00
Transactions on Pattern Analysis and Machine Intelligence (1351)	6	☐ \$ 7.50	☐ \$15.00
Transactions on Software Engineering (1171)	12	☐ \$ 9.00	☐ \$18.00
Journal on Lightwave Technology (4301)	12	☐ \$ 9.00	☐ \$18.00
Journal of Robotics and Automation (4401)	4	☐ \$ 5.00	☐ \$10.00
Journal of Solid State Circuits (4101)	4	☐ \$ 4.00	☐ \$ 8.00
Proceedings of the IEEE (5011) (available only to IEEE members)	12	☐ \$ 8.50	☐ \$17.00

☐ Diners Card
 ☐ VISA
 ☐ MasterCard
 ☐ American Express
 Mo Yr

☐ Check or money order enclosed
 \$

☐ Make payable to IEEE
 Total amount remitted with this application

☐ Check box for additional information:
 ☐ Student Membership Application
 ☐ Publications Catalog
 ☐ Bulk air option (two-to-six-week delivery of publications) for members in Western Europe, Latin America, and the Far East

I hereby make application for IEEE Computer Society membership and, if elected, will be governed by IEEE's and the Society's Constitutions, Bylaws, and Statements of Policies and Procedures.

MAILING ADDRESS

Full signature _____ Date _____

First Name _____ Middle initial(s) _____ Last name _____

Street address _____ City _____ State/Country _____ Zip _____

EDUCATION (highest level completed) _____ Course _____ Degrees received _____ Date _____

Name of educational institution _____ Date of birth _____ ☐ Male ☐ Female

OCCUPATION _____ TITLE or position _____ REFERENCE (an IEEE member, if unknown, a managerial person who knows you professionally) _____

Firm name _____ Name (print in full) _____ IEEE Member No. (if applicable) _____

Firm address _____ Position _____

City _____ State/Country _____ Zip _____ Street address _____

Affiliate society membership no. (if applicable) _____ Grade _____ City _____ State/Country _____ Zip _____

MAIL TO: IEEE COMPUTER SOCIETY, 10662 LOS VAQUEROS CIRCLE, LOS ALAMITOS, CA 90720-2578

tion and browsing capabilities. Since we anticipate that users may frequently specify content by means of some text words, we have discussed in depth some text access methods appropriate for this environment.

Several desirable features of multimedia object management systems are unique; they are not provided by traditional database management systems. Problems related to user interfaces, multimedia object structure, content addressability, and performance have to be studied through detailed design and experimentation. Integrated prototypes demonstrating the ideas should be developed. User feedback will help in pinpointing the limitations and improve the functionality of those systems. Finally, user acceptance of multimedia object servers will also depend on the overall system performance. Methods for improving system performance in a multimedia server environment should also be studied. □

References

- G. Ammon, G. Calabria, and D. Thomas, "A High-Speed, Large-Capacity, 'Jukebox' Optical Disk System," *Computer*, Vol. 18, No. 7, July 1985, pp. 36-45.
- A. Bell, "Critical Issues in High-Density Magnetic and Optical Storage," *Proc. SPIE*, Vol. 32, Optical Data Storage, Jan. 1983.
- L. Fujitani, "Laser Optical Disk: The Coming Revolution in On-Line Storage," *Comm. ACM*, Vol. 27, No. 6, June 1984, pp. 546-554.
- Kenney et al., "An Optical Disk Replaces 25 Mag Tapes," *IEEE Spectrum*, Feb. 1979, pp. 33-38.
- D. Maier, "Using Write-Once Memory for Database Storage," *Proc. ACM PODS*, 1982, pp. 239-264.
- S. Christodoulakis, "Multimedia Database Management: Applications and Problems, A Position Paper," *Proc. ACM SIGMOD*, Austin, Tex., May 1985, pp. 304-305.
- R. Nicholson, "Usage Patterns in an Integrated Voice and Data Communications System," *ACM Trans. Office Information Systems (TOOIS)*, Vol. 3, No. 3, July 1985.
- S. Christodoulakis, F. Ho, and M. Theodoridou, "The Multimedia Object Presentation Manager in MINOS: A Symmetric Approach," *Proc. ACM SIGMOD*, Washington, DC, May 1986.
- BNR, "Meridian System Information," *Telesis (BNR quarterly magazine)*, Vol. 12, No. 2, 1985.
- L. Koved and B. Shneiderman, "Embedded Menus: Selecting Items in Context," *Comm. ACM*, Vol. 29, No. 4, Apr. 1986, pp. 312-318.
- S. Christodoulakis et al., "Multimedia Document Presentation, Information Extraction, and Document Formation in MINOS: A Model and a System," *ACM Trans. Office Information Systems (TOOIS)*, article to appear.
- S. Christodoulakis, "Analysis of Retrieval Performance for Records and Documents," *ACM Trans. Database Systems (TODS)*, submitted.
- S. Christodoulakis, "Issues in the Architecture of a Document Archiver Using Optical Disk Technology," *Proc. ACM SIGMOD*, Austin, Tex., May 1985, pp. 34-50.
- S. Christodoulakis and T. Velissaropoulos, "Issues in the Design of a Distributed Testbed for MINOS," *Trans. Management Information Systems*, article to appear.
- D. E. Knuth, *The Art of Computer Programming*, Vol. 3 (Sorting and Searching), Addison-Wesley, Reading, Mass., 1973.
- J. Nievergelt, H. Hinterberger, and K. C. Sevcik, "The Grid File: An Adaptable, Symmetric Multitree File Structure," *ACM TODS*, Vol. 9, No. 1, Mar. 1984, pp. 38-71.
- R. S. Boyer and J. S. Moore, "A Fast String-Searching Algorithm," *Comm. ACM*, Vol. 20, No. 10, Oct. 1977, pp. 762-772.
- L. A. Hollaar, "Text Retrieval Computers," *Computer*, Vol. 12, No. 3, Mar. 1979, pp. 40-50.
- IBM, *STAIRS/VS: Reference Manual*, IBM System Manual.
- G. Salton and M. J. McGill, *Introduction to Modern Information Retrieval*, McGraw-Hill, New York, 1983.
- R. L. Haskin, "Special-Purpose Processors for Text Retrieval," *Database Engineering*, Vol. 4, No. 1, Sept. 1981, pp. 16-29.
- F. Rabiti and J. Zivka, "Evaluation of Access Methods to Text Documents in Office Systems," *Proc. Third Joint ACM-BCS Symp. Research and Development in Information Retrieval*, Cambridge, England, 1984.
- S. Christodoulakis and C. Faloutsos, "Design Considerations for a Message File Server," *IEEE Trans. Software Engineering*, Vol. SE-10, No. 2, Mar. 1984, pp. 201-210.
- C. Faloutsos, "Access Methods for Text," *ACM Computing Surveys*, Vol. 17, No. 1, Mar. 1985, pp. 49-74.
- G. Salton, *The SMART Retrieval System—Experiments in Automatic Document Processing*, Prentice-Hall, Englewood Cliffs, N.J., 1971.
- C. J. Van Rijsbergen, *Information Retrieval*, 2nd ed., Butterworths, London, England, 1979.
- R. A. Gustafson, "Elements of the Randomized Combinatorial File Structure," *Proc. ACM SIGIR Symp. Information Storage and Retrieval*, University of Maryland, College Park, Md., Apr. 1971, pp. 163-174.
- R. M. Bird, J. C. Tu, and R. M. Worthy, "Associative/Parallel Processors for Searching Very Large Textual Data Bases," *Proc. Third ACM Workshop on Computer Architecture for Non-numeric Processing*, New York, May 1977, pp. 8-16.
- L. A. Hollaar et al., "Architecture and Operation of a Large, Full-Text Information-Retrieval System," in *Advanced Database Machine Architecture*, D. K. Hsiao, ed., Prentice-Hall, Englewood Cliffs, N.J., 1983, pp. 256-299.
- C. M. Gravin, "National Westminster Bank Mass Storage Archiving," *IBM Systems J.*, Vol. 17, No. 4, 1978, pp. 344-358.
- C. Faloutsos, "Extending A DBMS to Handle Text," master's thesis, University of Toronto, Dept. of Computer Science, Oct. 1982.
- C. Faloutsos, "Signature Files: Design and Performance Comparison of Some Signature Extraction Methods," *Proc. ACM SIGMOD*, Austin, Tex., May 1985, pp. 63-82.
- D. Tschritzis and S. Christodoulakis, "Message Files," *ACM Trans. Office Information Systems*, Vol. 1, No. 1, Jan. 1983, pp. 88-98.
- P. A. Larson, "A Method for Speeding Up Text Retrieval," *Proc. ACM SIGMOD*, San Jose, Calif., May 1983.
- M. D. McIlroy, "Development of a Spelling List," *IEEE Trans. Communications*, Vol. COM-30, No. 1, Jan. 1982, pp. 91-99.
- S. Christodoulakis, "Access Files for Batching Queries in Large Information Systems," *Proc. ICOD II*, Aug. 1983.
- C. Faloutsos, "Integrated Access Methods for Messages Using Signature Files," *IFIP Conf. Methods and Tools for Office Systems*, Pisa, Italy, Oct. 1986.



Stavros Christodoulakis obtained his BSc in physics and mathematics from the University of Athens, Greece; his MSc in computer and information sciences from Queen's University, Canada; and, in 1981, his PhD in computer science from the University of Toronto.

He was an assistant professor at the University of Toronto from 1981 to 1985. He is currently with the Dept. of Computer Science of the University of Waterloo.

His interests include databases, multimedia servers, distributed systems, office automation, and performance evaluation.



Christos Faloutsos received the BSc degree in electrical engineering in 1981 from the National Technical University of Athens, Athens, Greece. He received the MSc and PhD degrees in computer science from the University of Toronto, Toronto, Canada.

Since 1985, he has been an assistant professor in the Dept. of Computer Science at the University of Maryland, College Park.

His research interests include physical database design, information retrieval, office automation, and performance evaluation.

Readers may write to Stavros Christodoulakis at the University of Waterloo, Faculty of Mathematics, Dept. of Computer Science, Waterloo, Ontario N2L 3G1, Canada. They may also write to Christos Faloutsos at the University of Maryland, Dept. of Computer Science, College Park, MD 20742.

IEEE INFOCOM '87

The Conference on Computer Communications

PREVIEW PROGRAM

Sixth Annual Conference
Global Networks • Concept to Realization

Tutorials: March 30, 1987
Technical Sessions: March 31-April 2, 1987

CONFERENCE COMMITTEE

General Chair: Howard A. Blank, Booz-Allen & Hamilton, Inc.
Vice Chair: Celia L. Desmond, Telecom Canada
Program Chair: Izhak Rubin, U.C.L.A.

*Meridien Hotel
San Francisco, California*



THE COMPUTER SOCIETY
OF THE IEEE



THE INSTITUTE OF ELECTRICAL
AND ELECTRONICS ENGINEERS, INC.



IEEE COMMUNICATIONS SOCIETY

Technical Sessions

IEEE INFOCOM'87

Tuesday, March 31, 1987

**9:30—
10:45 am** Keynote Session

11:00 am — 12:30 pm	Session T1A Multiple-Access I	Session T1B Network Standards	Session T1C Distributed Resource Allocation & Control #1	Session T1D Network Management Modeling & Support
2:00— 3:30 pm	Session T2A Local Area Networks I	Session T2B Routing	Session T2C Rapidly Reconfiguring Networks	Session T2D End-to-End Network Management
4:00— 5:30 pm	Session T3A Random Access	Session T3B Fiber Optic Networks	Session T3C Distributed Resource Allocation & Control II	Session T3D Panel: Satellite Networks and ISDN

Wednesday, April 1, 1987

8:30— 10:00 am	Session W1A Multiple-Access II	Session W1B Metropolitan Area Networks	Session W1C The Application of Expert Systems in Network Operations
10:30— 12:00 Noon	Session W2A Local Area Networks II	Session W2B Performance Analysis for Error Control Schemes	Session W2C Network Services
1:30— 3:00 pm	Session W3A LAN Experimentation and Implementation	Session W3B Network Reliability	Session W3C Queue Control for Computer Communications
3:30— 5:00 pm	Session W4A Military Communications	Session W4B Design and Analysis of Communication Protocols	Session W4C Panel: Future Directions in Computer Communications

Thursday, April 2, 1987

8:30— 10:00 am	Session TH1A Local Area Networks III	Session TH1B Communications Switching	Session TH1C Distributed Processing
10:30— 12:00 Noon	Session TH2A Integrated Packetized Voice and Data Networks	Session TH2B Packet Radio and Mobile Radio Networks	Session TH2C Panel: IBM's Open Network Management Architecture
1:30— 3:00 pm	Session TH3A Satellite Communications Networks	Session TH3B Queueing Analysis	Session TH3C Network Management Local Area Networks
3:30— 5:00 pm	Session TH4A Network Implementations	Session TH4B Performance Evaluation	Session TH4C High Speed Network Architectures

On Integrated Networks for Diverse Applications

Dr. Jonathan S. Turner
Washington University

Objective

The next generation of large public communications networks need to be largely independent of specific applications and be capable of incorporating new technologies. This tutorial will examine alternatives for such future networks, including fast circuit switching, fast packet switching and one or two hybrid approaches. It will then focus on fast packet switching which seems to be the most promising approach.

Audience

This tutorial will be useful to data communications personnel who need to learn about high speed switching and anyone interested in the evolution of communication technology.

Outline

1. Introduction and motivation
2. Review of switching technology
3. Fast packet switching
4. Multi-point communication
5. Research review

Instructor

Jonathan S. Turner is an Associate Professor of Computer Science at Washington University, St. Louis, MO. He received his Ph.D. from Northwestern University and until 1983 was with Bell Laboratories. He has conducted research on high-performance communications systems, coping with hard computational problems and integrated computation and communication systems.

Local Area Networks for Real-Time Applications

Dr. Gerard Le Lann
SCORE

Objective

Distributed computing and real-time computing technologies have evolved separately until recently, but must be integrated for computing systems used in industrial and defense applications with severe real-time and reliability requirements. The tutorial will concentrate on classes of traffic, physical architectures, transmission media, multiple access protocols, message scheduling algorithms, end-to-end protocols, performance evaluations, standardization trends and VLSI circuits in the context of distributed computing systems implemented on real-time local area networks.

Audience

This tutorial should be of interest to distributed processing project managers, system architects, researchers, developers, software engineers, and users of real-time industrial/defense systems.

Outline

1. Coaxial cables and optical fibers
2. Circular vs linear communications media
3. Multiple-access protocols
4. The IEEE 802 Committee Standards
5. Case studies (Ethernet, Starlan, Token-ring)
6. Requirements of real-time applications
7. Classes of real-time local area networks
8. Case studies (Bitbus, Map/One, MVM 370/372, i80C152)

Instructor

Dr. Gerard Le Lann is Director of SCORE (INRIA), a project aimed at designing and prototyping real-time distributed computing systems intended for fully automated environments. Previously he conducted research in distributed databases, real-time shipborne multiprocessor system, local area networks and wide-area computer communication networks. He holds a diploma d'Ingenieur en Informatique and a Ph.D. in computer science.

Network Interconnection and Gateways

Dr. Carl Sunshine
SDC

Objective

The proliferation of personal computers and intelligent workstations is causing an explosion in the number of computer networks. This tutorial will focus on technological problems of interconnecting networks, present possible solutions, and indicate likely developments for research, commercial, and military systems.

Audience

The tutorial is intended for programmers, systems analysts, engineers, and managers involved with computer network development or planning, and for network users desiring indepth understanding. Some background in networks and communication protocols is desirable.

Outline

1. Introduction
2. Theory (Strategies, Interconnection, Routing, Flow, Fragmentation)
3. Summary of gateway functions
4. Political and regulatory issues
5. Case studies
6. Advanced issues

Instructor

Carl Sunshine helped develop the ARPA internet protocols during his studies at Stanford University where he completed his Ph.D. in 1975. Since then he has worked at the Rand Corp., USC's Information Sciences Institute and Sytek. Currently, Dr. Sunshine is Chief Network Architect at SDC, where he is responsible for design and development of SDC's network systems for DoD and other government customers.

Electronic Document Representation and Standards

Dr. Najah Naffah
Bull Transac

Objective

Electronic systems such as electronic mail, electronic publishing, teleconferences and videoconferencing will exchange and manipulate multimedia documents (text, graphics, voice, images, etc.) in the future. The tutorial will focus on several current proposals for describing the multimedia document presentation and review their prominent features and relative advantages.

Audience

The tutorial is aimed at managers, engineers, and software designers involved or interested in planning, design, and development of multimedia data systems.

Outline

1. Office environment
2. Current and future applications
3. Open system architecture
4. Current ISO/standards and proposals
5. Current CCITT proposals
6. Industry and research proposals
7. Printing standards
8. Examples of systems implementations
9. Conclusions

Instructor

Najah Naffah is head of the advanced studies department in office systems at Bull Transac, where he manages a group of researchers in various areas such as workstation design, document production, database management systems, and artificial intelligence-based applications. Prior to this, he was a leader of the Kayak project at INRIA-Rocquencourt in France and a researcher for Cyclades, a datagram-based computer network. He holds a Ph.D. in computer science from the University of Paris.

Hotel Meridien Registration

Reservations must be received by the Hotel Meridien by February 27, 1987

Complete this form and return to

HOTEL MERIDIEN
50 Third Street
San Francisco, CA 94103
(415) 974-6400

Please check accommodations and price desired:

- ☐ Single occupancy ☐ \$95 ☐ \$115 ☐ \$135
☐ Double occupancy ☐ \$115 ☐ \$135 ☐ \$155

Arrival Date _____ Departure Date _____
☐ am ☐ pm ☐ am ☐ pm

All major credit cards accepted

- ☐ Visa ☐ Master Card ☐ Carte Blanche ☐ Diners ☐ American Express

Card No. _____ Exp. Date _____ Signature _____

Name _____

Address _____ City/State/Zip/Country _____

To confirm a reservation, a \$125.00 advance deposit is required. This deposit is refundable if reservation is cancelled with 48 hours notice.

Advance Registration

Complete this form and return with check (U.S. Dollars) or credit card endorsement to:

IEEE INFOCOM '87
IEEE Computer Society
1730 Massachusetts Avenue, NW
Washington, DC 20036-1903
(202) 371-0101

ADVANCE REGISTRATION prior to 3/14/87

LATE REGISTRATION after 3/14/87

Please check appropriate box

- ☐ **Tutorial 1** On Integrated Networks for Diverse Applications
☐ **Tutorial 2** Network Interconnection and Gateways
☐ **Tutorial 3** Local Area Networks for Real-Time Applications
☐ **Tutorial 4** Electronic Document Representation and Standards
☐ **Conference Only**
☐ **Tutorial 1 and Conference**
☐ **Tutorial 2 and Conference**
☐ **Tutorial 3 and Conference**
☐ **Tutorial 4 and Conference**

- ☐ Student Conference Only, must be IEEE Student
Member, not employed full time and show membership
card at door; does not include proceedings.

Member	Non-Member	Member	Non-member
\$180	\$220	\$210	\$260
180	220	210	260
180	220	210	260
180	220	210	260
360	440	420	520
360	440	420	520
360	440	420	520
360	440	420	520
100		125	

Requests for refunds must be received in writing NO LATER THAN March 14, 1987. TUTORIAL registrations include luncheon, notes and coffee breaks. CONFERENCE registrations include one copy of the proceedings, conference wine & cheese Tuesday and Wednesday evenings.

Name _____ Company Name _____

Street Address _____ City _____ State _____ Zip _____

- ☐ IEEE ☐ IEEE Computer Society ☐ IEEE Communications Society

Member No. _____ Spouse Name (if attending) _____

- ☐ Check Enclosed ☐ Visa ☐ MasterCard ☐ American Express

Card No. _____ Exp. Date _____ Signature _____

THE OPEN CHANNEL

"Any clod can have the facts,
but having opinions is an art."

Charles McCabe,
San Francisco Chronicle

Let's teach real-world programming

It is time for our computer science departments to change the way they teach programming. Most schools teach programming by emphasizing the creation of complete computer programs. This approach is much too limited for many reasons, but the big ones are that it inherently emphasizes small problems and rarely teaches anything about programming teamwork.

In the real world, the new programmer is going to discover that most important problems involve existing programs and that he seldom has the freedom to completely rewrite those programs. In the real world, the programs bear the footprints of every programmer who has wandered through the code, and in spite of the increasing amount of lip service being paid to documentation at every level, real programmers don't document their code very well. If the programmer is lucky, he is going to get the opportunity to work on big programs, much bigger than he could code by himself within a semester. The lucky programmer will suddenly find himself within a team of programmers who are trying to divide and conquer some horrendous problem.

There are lots of methods our computer science departments can use to help train new programmers to face the real world. Most of them involve making the student work with other students and with existing programs. So here are a few sample ideas to help get some thinking started.

One of the more commonly used approaches is to assign larger problems and divide the students into programming teams. Unfortunately, what often happens (in my limited experience) is that

the best programmer in the team winds up writing the entire project. Obviously, this method needs to be used with some caution.

Or how about this idea for exposing students to good programming style? Give the students programs that are good examples of how to write code and assign program modifications to solve new problems, preferably without destroying the programs' existing functions. This would force students to learn to read more complex code and help them learn what features make code readable—crucial skills that I somehow missed way back when.

Another approach would be to assign each student a small program and give the finished program to another student for modification. The first student's grade would be affected by how easily the second student could modify the program. This approach would help students appreciate what makes some code easy to understand, work with, and modify, as well as teaching them the importance of being able to find the original programmer. (I don't suppose I'm the only one who's had to clean up after a programmer who's left the company and moved on to who knows where—possibly Brazil.)

For advanced classes, students should have to practice making some of the harder programming decisions—for example, how and when to implement various features when you know in advance that certain significant changes will be made to the programs. Of course, in the real world, the programmer usually doesn't know what kind of changes will be required, but a careful professor

could select the kinds of changes commonly called for, thus helping students get a feeling for where programs need "to bend." As a trivial example, I need only mention all the programming time saved by knowing when a constant is really a parameter.

These are just some ideas for things that my 20/20 hindsight shows would have been helpful. The educational limitations these ideas seek to address are important both in academia and the harsh commercial world. Some of these approaches are already being used to various degrees, but methods like these need to become much more prevalent than they were when I was in the university.

Shannon Jacobs
Austin, Texas

This reminds me of a conversation with Bill McKeeman about a course taught at the University of Toronto in 1972. (See James J. Horning and David B. Wortman, "Software Hut: A Computer Program Engineering Project in the Form of a Game," IEEE Transactions on Software Engineering, July 1977, pp. 325-330.) The entire class was assigned a single large programming project divided into two major modules. The class was divided into teams of three members each, and the rules went like this:

- (1) Each team is to produce one of the modules.
 - (2) Each team is to write a driver program and integrate the two modules purchased from other teams into a system.
 - (3) Each team is to purchase a complete system not containing its own module from another team and modify it according to revised specifications.
 - (4) Each team's grade is partially determined by its success in selling its module and system to the other teams.
- Bill is now with the Wang Institute of Graduate Studies, Tyngsboro, Massachusetts, which uses a highly evolved descendent of the original course.

J.H.

The Open Channel is exactly what the name implies: a forum for the free exchange of technical ideas. Try to hold your contribution to one page maximum in the final magazine format (about 1000 words).

We'll accept anything (short of libel or obscenity) so long as it's submitted by a member of the Computer Society. If it's really bizarre we may require you to get another member to cosponsor your item.

Send everything to Jim Haynes, Computer Center, UC Santa Cruz, CA 95065.
Bitnet: haynes@ucsc; Arpa: ucsc@haynes@ucvax.berkeley.edu.

UPDATE

Computer Society election results announced

The IEEE has announced the results of Computer Society balloting for four elective offices, 10 open positions on the Board of Governors, and two proposed constitutional amendments. A total of 66,896 ballots were sent by first-class mail to Computer Society members in mid-September; 10,080 (15.1 percent) were returned to IEEE headquarters by the October 20 deadline. This is a slight increase over the 14.3-percent return in last year's election when ballots were sent by third-class mail.

Officers. Votes received by all candidates for Computer Society office are given below. The individuals elected will serve one-year terms beginning January 1, 1987.

President:		
Roy L. Russo (elected)	9034	
President-elect:		
Taylor L. Booth	4733	
Edward Parrish (elected)	4831	
First vice president:		
James Ayler	3650	
Michael C. Mulder (elected)	5741	
Second vice president:		
Kenneth R. Anderson (elected)	5096	
John D. Mosa	4329	

Board of Governors. The 10 individuals receiving the highest number of votes are elected to two-year terms beginning January 1, 1987.

Tilak Agerwala	3366
Mario R. Barbacci (elected)	3657
Victor R. Basili (elected)	4118
P. Bruce Berra	3523
Wushow Chou	3524
Lorraine M. Duvall (elected)	6407
Michael Evangelist (elected)	3772
Wolfgang K. Gili	2666
Allen L. Hankinson (elected)	3950
William Howden	3094
Barry Johnson	3394
Laurel Kaleda (elected)	4629
Ted Lewis (elected)	4876
Ming T. Liu (elected)	4206
H. Troy Nagle, Jr.	3559
Raymon Oberly	3589
Gordon C. Padwick	2836
Roland J. Saam	3141

Earl E. Swartzlander, Jr.
(elected) 4821
Joseph Urban (elected) 4329

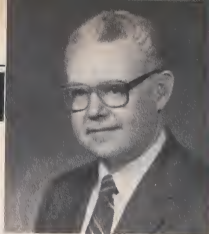
Constitutional amendments. Both of the proposed constitutional amendments received the required two-thirds favorable vote and passed. The vote on the amendment to change the constitutional reference to the society's retiring president from "junior past president" to "immediate past president" was 8929 votes to approve and 166 votes to disapprove. The amendment modifying the definition of majority vote, to prevent an abstention from acting as a negative vote, received 8422 votes for adoption and 592 votes against.

Programs granted CSAB accreditation

The Computing Science Accreditation Board recently accredited 22 computer science programs, the first accreditations granted since the board's 1984 formation by ACM and the IEEE Computer Society. The CSAB granted accreditation to 22 of the 31 four-year institutions that had applied for it. In the 1986-87 cycle, the board expects to accredit 35 to 40 additional programs.

Accreditation is based on an evaluation of how well the program meets general academic guidelines and how well it meets the institution's goals in establishing the program.

The accredited programs are offered by Allegheny College, Brandeis University, California State Polytechnic University at San Luis Obispo, California State University at Sacramento, Clemson University, Drexel University, Georgia Institute of Technology, Iowa State University, Mississippi State University, New Jersey Institute of Technology, North Dakota State University, North Texas State University, Northeastern University, Pace University, Stevens Institute of Technology, University of California at San Diego, University of California at Santa Barbara, University of Missouri at Rolla, University of Utah, US Air Force Academy, Western Michigan University, and Worcester Polytechnic University.



Taylor L. Booth, 53, dies

Taylor L. Booth, a computer science and engineering professor at the University of Connecticut and director of the university's Computer Applications and Research Center, died of a heart attack October 20. Booth, who was 53, is survived by his wife, Aline, three children, and two grandchildren.

Active in the Computer Society for over 16 years, particularly in its education activities, Booth was instrumental in defining computer science and engineering curricula for program accreditation through the society's and the IEEE's respective boards. He also worked for closer cooperation between the Association for Computing Machinery and the IEEE Computer Society.

The Computer Society's Board of Governors, meeting November 7 at the conclusion of the ACM/IEEE-CS Fall Joint Computer Conference, cited Booth for his leadership and contributions in the education field. The resolution, introduced by *Computer Editor-in-Chief* Michael C. Mulder, praised Booth for "dedication, completeness, and a special sense of fairness that provides a model for those who will follow him."

At the time of his death, Booth was serving as chairperson of the Computer Society's Constitution and Bylaws Committee and was a member of its Educational Activities Board. Previously, he had served as first vice president, secretary, vice president for educational activities, and member of the Board of Governors. He had been editor-in-chief of *IEEE Transactions on Computers*, a distinguished visitor, and a founding member and the first president of the Computing Science Accreditation Board.

An IEEE fellow, Booth was an IEEE representative to the Engineering Accreditation Commission and an alternate member of the Accreditation Board for Engineering and Technology. He was also a member of the IEEE Education Activities Board.

Booth received the Frederic Emmons Terman Award in 1972 and the IEEE Centennial Medal in 1984.

1987 IEEE INTERNATIONAL SYMPOSIUM ON INTELLIGENT CONTROL

January 18 to 20 1987 - Hilton Hotel - Philadelphia - Pennsylvania

Sponsored by The IEEE Computer Society. Co-sponsored by The IEEE Control Systems Society,
The Philadelphia Section of IEEE. In cooperation with The IEEE Council on Robotics and Automation.
Hosted by Drexel University.

General Chairman: A. Meystel, Drexel University. (215- 895-2220)
Program Chairman: J.Y.S. Luh, Clemson University. (803-656-5926)
Special Presentations: G. Saridis, RPI. L. Zadeh, UC at Berkeley.

Arrangements: N.M. Bilgutay
Treasurer: U. Bar-Am
Publication: P. Kalata
Publicity: J. Eilbert
Registration: P. Rao



Topics of Papers:

- Control of Stochastic and Distributed Systems
- Information Structures for Intelligent Control
- Adaptive and Self-organizing Controllers
- Intelligent Controllers for Large Space Systems
- Cognitive Controllers for Autonomous Systems
- Computer Architectures for Intelligent Control
- Controllers with Associative Memory
- Control Systems with Quantitative and Conceptual Learning
- Multiple Agent Systems: Cooperative and Antagonistic Control
- Structures of Intelligent Control
- Neural Networks as a Control Tool
- Multiple Sensor Feedback Controllers
- Linguistical Controllers
- Fuzzy Control Systems
- Unmanned Control Systems
- Knowledge Based Control Systems
- Team Controller

INVITED SESSIONS
Hybrid Knowledge-Based / Analytical Control Systems
Multilayer Controllers for Large Systems
Intelligent Control Systems for Assembly Robots

Advisory Committee

Albus, J. National Bureau of Standards, USA
Bajcsy, R. University of Pennsylvania, USA
Bejczy, A.K. Jet Propulsion Laboratory, USA
Chen, C. Jiao-Tong University, Shanghai, PRC
Clarke, D.W. University of Oxford, England
Di Cesare, F. Rensselaer Polytechnic Institute, USA
Furuta, K. Tokio Institute of Technology, Japan
Goldenberg, A.A. University of Toronto, Canada
Heer, E. E.Heer & Associates, USA
Ho, Y.C. Harvard University, USA
Kanade, T. Carnegie-Mellon University, USA
Koivo, A. Purdue University, USA
Landau, I.D. Lab d'Automatique de Grenoble, France
Rembold, U. University of Karlsruhe, Germany
Saridis, G. Rensselaer Polytechnic Institute, USA
Stephanou, H. G. Mason University, USA
Villa, A. Politecnico di Torino, Italy
Weissbin, C. Oak Ridge National Laboratory, USA

REGISTRATION

On/before Jan 5, 1987			After Jan 5, 1987			Cancellation fee
Student	IEEE-Member	Other	Student	IEEE-Member	Other	
\$50	\$175	\$230	\$70	\$210	\$275	\$20

Note -- Fee includes (except for students) lunch, coffee & pastries, social events and a copy of the proceedings.

Student fee covers attendance of sessions and a copy of the proceedings only.

Payment -- In US dollars only; by check; payable to IC-87. Send check and registration form to: Dr. Prabhakar Rao,
Intelligent Control-1987, Department of ECE, Drexel University, Philadelphia, PA 19104.

Name (First) _____ (Last) _____
Affiliation _____
Student ID Number (if applicable) _____
Address _____
City _____ State _____ Zip Code _____ Country _____

THE HILTON OF PHILADELPHIA ROOM RESERVATION

Name (First) _____ (Last) _____ Telephone _____
Address _____
City _____ State _____ Zip Code _____ Country _____
Arrival Date _____ Departure Date _____
Single (\$65.00) _____ Double (\$75.00) _____ EXP _____
Credit Card (AX, CB, DC, VISA, MC) _____ # _____

To guarantee a reservation, please forward a one night's deposit or include your valid credit card number with expiration date.

Any reservations without a guarantee will only be held until 6:00 p.m. on date of arrival. Please return to:

THE HILTON OF PHILADELPHIA, Civic Center Boulevard at 34th str., Philadelphia, PA, 19104



THE COMPUTER SOCIETY
OF THE IEEE



THE INSTITUTE OF ELECTRICAL AND
ELECTRONICS ENGINEERS, INC.

NEW PRODUCTS

Compaq adds new Deskpro 386 model

Compaq Computer has added the Model 70 to its Deskpro 386 line. The Model 70 features a 70M-byte fixed disk drive, a 1.2M-byte disk drive, a 1M-byte RAM, and five expansion slots. It costs \$7299.

The company also announced that the Compaq color monitor (\$799) and the Compaq enhanced color graphics board (\$599) are now available as options for the entire Compaq product line. Moreover, the enhanced keyboard used on all models of the Deskpro 386 will become standard on the Deskpro 286.

The 80287 coprocessor option (\$349) is now available for all 80286-based Compaq products, including the Portable 286, Deskpro 286, and Portable II. The Portable II line also has an optional third-height, 5.25-inch, 1.2M-byte disk drive now available for \$275.

For more information, contact Compaq Computer Corp., 20555 FM149, Houston, TX 77070; (713) 370-0670.

Reader Service Number 40

Sperry expands with computers and software

Sperry has announced a number of computer products, including the 2200/200 entry-level midframe system. The four basic models are the 2200/201, a uniprocessor and entry-level system; the 2200/202, a dual processor system; the 2200/203, a three-processor model that requires an expansion cabinet; and the 2200/204, a four-processor system.

The 2200/200 systems are compatible with the OS 1100 environment. They also have Shield software, a PC-based operator interface, and support such programming languages as Cobol, Fortran, RPGII, and MASM.

An entry-level 2200/201 with one processor, 4M bytes of memory, disk capacity of 840M bytes, a 300 line-per-minute printer, and communications capability costs \$198,026. A two-processor 2200/202 with 8M bytes of storage, disk capacity of 1872M bytes, a 640 line-per-minute printer, a Uniservo 22 tape drive, and a DCP/110A communications processor costs \$404,180.

The Mapper C system is Sperry's newest version of its fourth-generation language, Mapper. It runs under the Unix system. A one-time license fee ranges from \$7900 to \$42,000 according to the host system.

For more information on the new Sperry products, contact Sperry Corp., World Headquarters, Blue Bell, PA 19424-0031; (215) 542-4213/4217.

Reader Service Number 41

Wang offers LapTop computer

Wang Laboratories has announced the LapTop computer, a 14.25-pound portable featuring 10M-byte hard disk storage, LCD display, a full-sized keyboard, built-in communications and printer, and IBM compatibility using the MS-DOS 3.2 operating system.

The battery-driven LapTop includes a 8086-compatible 16-bit CMOS microprocessor operating at 8 MHz; 512K-bytes of RAM expandable to 1M-byte; 16 programmable function keys; optional floppy drives for 3.5-inch, 720K-byte, and 5.25-inch disks; either a 300/1200 or 300/1200/2400 baud Hayes-compatible modem; IBM PC-XT compatibility; and IBM color graphics adapter compatibility.

The LapTop costs \$3530. For more information, contact Wang Laboratories, Inc., One Industrial Ave., Lowell, MA 01851; (617) 459-5000.

Reader Service Number 42

First programs out from Lotus ESPD

Lotus Development has announced the first two products offered by its Engineering and Scientific Products Division. Lotus Manuscript is a word and document processor, while Lotus Measure is a PC software package that collects data from measurement hardware and puts it into Lotus 1-2-3 for analysis, graphic display, and storage.

Manuscript provides word processing capabilities of cut and paste, copy, search and replace, and form-letter generation. It also features table handling, a built-in structured outline, global indexing and footnoting, a spelling checker, a document comparison feature that highlights revisions, and a document preview capability.

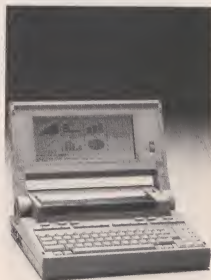
Manuscript's suggested price for North America is \$495.

Measure works with 1-2-3 as a single program. It employs the same user interface and macro environment. It supports IEEE-488 and RS-232-C communications. The program runs on the IBM PC, PC-XT, or PC-AT, the HP Vectra PC, and the Compaq portable. It requires 512K bytes of memory and a hard disk. It runs on MS-DOS version 2.0 or higher, and with Lotus 1-2-3 version 2.0 or higher.

Measure is priced at \$495 in North America.

Contact Lotus Development Corp., 55 Cambridge Pkwy., Cambridge, MA 02142; (617) 577-8500.

Reader Service Number 43



The Wang LapTop computer.

IBM midrange computers will extend System/370

IBM has announced the IBM 9370 Information System, which consists of four computers: the entry-level Model 20, intermediate-level Models 40 and 60, and high-end Model 90. Field testing is scheduled for February 1987 and general availability for the fourth quarter of 1987.

Prices for the new computers will range from \$31,000 for the Model 20 with 4M-bytes of memory to \$210,000 for the Model 90 with 16M-bytes of memory. The IBM 9347 Magnetic Tape Unit will cost \$7900.

I/O controllers for the 9370 Information System include a DASD/tape subsystem controller, a workstation subsystem controller, four communications subsystem controllers, and an optional System/370 block multiplexer channel.

Software includes the VM/Integrated System, VM/Remote System Programming, VM/System Product Release 5, VSE/SP V.3 Release 1, and Decision and Information Support Productivity Facility for VSE (optional).

Processors have a maximum memory capacity of 16M bytes. Model 20 comes with 4, 8, or 16M bytes. Models 40, 60, and 90 come with 8 or 16M bytes.

For more information, contact IBM Corp., Information Systems Group, 900 King St., Rye Brook, NY 10573; (914) 934-4488.

Reader Service Number 44

Products link Mac and DEC

Odesta and Kinetics have announced software and hardware products linking the Apple Computer and Digital Equipment standard networks.

Odesta's Helix VMX software is a data-based, multiuser applications environment designed to run on a DEC VAX through Apple Macintosh PCs. Connections between computers in the system are made over an AppleTalk network or multiple internets connected to an Ethernet backbone. The program can be purchased per user fees or per VAX license. The former costs \$450 for each Macintosh connected to any VAX. The latter starts at \$7500 for a MicroVAX II and ranges to \$20,000 on larger VAX machines. The bridge to Ethernet, FastPath, is available from Odesta for \$2500.

Kinetics' FastPath links Apple Computer and DEC standard networks. FastPath/Stand-alone is a programmable gateway between AppleTalk and Ethernet. It allows computers on each of the networks to participate in common applications. It costs \$2500. FastPath/Q-Bus is a dual-height AppleTalk controller board for Q-bus backplanes. It allows the MicroVAX to participate in an AppleTalk network alongside Apple computers and LaserWriters. It lists for \$1800.

For more information on Helix VMX, contact Odesta Corp., 4084 Commercial Ave., Northbrook, IL 60062; (312) 498-5615.

For more information on FastPath, contact Kinetics, Inc., 2500 Camino Diablo, Suite 110, Walnut Creek, CA 94596; (415) 947-0998.

Reader Service Number 45

Supermini hits 1 MIP

Concurrent Computer has announced the Model 3212, a new member of its Series 3200 supermini computer family. According to the company, the new model provides real-time performance of up to 1.8 million Whetstone instructions per second using an optimized Fortran compiler.

Based on the Model 3210, the 3212 supports the company's proprietary operating system, OS/32, and the Xelos OS, based on AT&T's Unix System V, Release 2. The computer is compatible with the entire Series 3200 product line, according to Concurrent.

A basic configuration consists of 4M bytes of memory, eight communications ports, a line printer interface, and a Model 6100 video display unit. It also includes 1K byte of cache memory. The basic system costs \$42,000.

An OEM version, the Model 3212/A, allows for custom configuration.

For more information, contact Concurrent Computer, 197 Hance Ave., Tinton Falls, NJ 07724; (201) 758-7000.

Reader Service Number 46

Convex has Extended Supercomputing Architecture

Convex Computer has announced CXS, Extended Supercomputing Architecture, to allow multiple Convex supercomputers to work simultaneously but independently on multiple applications. Each node is a Cray-like computer system with dedicated memory bandwidth and I/O resources for each CPU. The architecture uses a fiber-optic interconnect and provides for modular growth.

Global mass storage permits sharing of data at the file level regardless of physical location. Two new members using CXS extend the C1 family of supercomputers at the top and bottom. The C1 XL is an entry-level supercomputer with a starting price of \$350,000. It

provides up to 64M bytes of memory, has an 80M-byte-per-second I/O bandwidth, and is compatible with the Convex C-1. The C1 XP offers the C1 family's greatest expandability. Pricing starts at \$475,000. Multiple-processor configurations bundling the CXS, the XP2 and XP4, are available starting at \$775,000 and \$1.35 million, respectively. To add the hardware element of CXS to other C1 family members costs \$25,000 to \$30,000 per CPU.

For more information, contact Convex Computer Corp., 701 Plano Rd., Richardson, TX 75023; (214) 952-0226.

Reader Service Number 47

Enhancements to Prolog

Systems Designers has announced SD-Prolog, a Prolog programming environment running on the IBM PC. The language is an extended version of the standard Prolog as defined by Clocksin and Mellish. It features multicolored windows, modular programming, DEC-10 box model debugging, context-sensitive on-line help, an integrated screen editor, pull-down menus, external interfaces (including C), and support for strings and floating-point arithmetic. According to the company, the language is based on an incremental compiler. A single site license costs \$799.

For more information, contact Systems Designers plc, Pembroke House, Pembroke Broadway, Camberley, Surrey GU15 3EX; (0276) 686200.

Borland International has enhanced Turbo Prolog. Version 1.1 provides increased support for the development of large applications, according to the company, and delivers a three-fold increase in compilation speed over Version 1.0. An internal linker with single-step compiling to executable files reputedly cuts development time.

For more information, contact Borland International, 4385 Scotts Valley Dr., Scotts Valley, CA 95066; (408) 438-8400.

Reader Service Number 48

32-bit computer fits on a desktop

Digital Electronic Systems has announced the Computerist Mainframe. According to the company, the CMF is a desktop, 32-bit, general-purpose computer system. It can expand into a multiuser, multiprocessor, multitasking real-time system in a standard 19-inch rack.

CMF provides a 32-bit address bus and up to 113M bytes of memory within the chassis. Expansion racks reputedly extend RAM to 4 billion bytes. The operating system is CP/M-68k, with optional languages available.

The basic CMF system costs \$3995. Contact Digital Electronics Systems, 302 S. Main, Estill Springs, TN 37330; (615) 649-5137.

Reader Service Number 49

TI expands series

Texas Instruments has announced a multiprocessor version of the Explorer computer system that combines symbolic and numeric processing in the same chassis. The Explorer LX system links Lisp and 68020 processors with a set of software tools. A field-installable upgrade is available for existing Explorer customers.

The base configuration consists of an Explorer with 4M-bytes of memory, a separate 68020 processor with 2M bytes of on-board memory, two 182M-byte disks, and an Ethernet interface or cartridge tape. Prices start at \$73,900. The upgrade costs \$18,500.

New storage devices for the Explorer systems include a 516M-byte Winchester disk drive and a half-inch magnetic tape drive. A 516M-byte disk drive with controller costs \$20,000; a 516M-byte disk drive add-on, \$17,000; and a half-inch tape drive, \$10,900.

For more information, contact Texas Instruments, Data Systems Group, PO Box 809063, Dallas, TX 75380-9063; (800) 527-3500.

Reader Service Number 50

Apollo expands Domain

Apollo Computer has expanded its Domain Series 3000 Personal Workstation product line with graphics, memory, and mass storage enhancements.

The new enhancements include 8M bytes of memory, 155M-byte or 348M-byte (formatted) Winchester disks, and a 19-inch color display. New communications options for the IBM PC-AT-compatible peripherals bus include the Serial/Parallel Expansion board; the EtherController-AT, providing Domain users with access to other systems through Ethernet networks and communications protocols; and Domain/PC, an IBM PC-AT coprocessor.

For information on how the enhancements and options affect the prices of the systems, contact Apollo Computer, 330 Billerica Rd., Chelmsford, MA 01824; (617) 256-6600.

Reader Service Number 51

NCR adds models

NCR has announced new models of its 80286-based PC8 PC and its Tower supermicrocomputer family.

Both new PC8 models use an 8-MHz Intel 80286 processor and come standard with 640K bytes of memory and a 20M-byte fixed disk, seven expansion slots, AT-card compatibility, and five media slots. One model, with a 360K-byte flex disk drive, is priced at \$3895, while the second, with a 1.2-MB flex disk drive, is priced at \$3995.

The Tower 32/400 is an entry-level, 32-bit model in the Tower supermicrocomputer family. It uses a 16.7-MHz Motorola MC68020 microprocessor and the Unix operating system. The list price ranges from \$14,995 to \$54,810, depending on configuration.

For more information, contact NCR Corp., Public Relations, Dayton, OH 45479; (513) 445-4169.

Reader Service Number 52

Minis join M6000 line

McDonnell Douglas Computer Systems has announced the addition of two minicomputers to the upper end of its M6000 line of small business systems.

Both the M6640 and M6650 offer from 16 to 64 ports and from 1-2M bytes of main memory. The M6650 comes with a basic 75M bytes of memory and two additional 130M-byte disk drives for a total of 335M bytes. The M6640 comes with 75M bytes of memory, field-upgradable with the addition of two 130M-byte disk drives. Maximum memory capacity for both systems is 485M bytes.

The M6000 line executes the Pick-based Reality relational database management operating system. The systems can also utilize McDonnell Douglas' ALL application language liberator, a fourth generation language.

Prices of the M6640 begin at \$59,500. The M6650 base price is \$84,500.

For more information, contact Gary Shoup, McDonnell Douglas, PO Box 19501, Irvine, CA 92713.

Reader Service Number 53

SNAs link to LANs

Eastman Communications' Syncra LAN Server offers remote connections or a channel speed adapter to link all computers on an Ethernet local area network to an SNA mainframe. The product is available from Eastcom through an agreement with Mitek Systems.

The Syncra LAN Server supports up to 64 concurrent terminal sessions. It appears as a 3274 cluster controller to the VTAM mainframe communications software. A variety of communications, emulation, and software options are available for varying prices. Contact Eastman Communications, 1099 Jay St., Rochester, NY 14650; (716) 724-5535.

Reader Service Number 54

Nixdorf adds midrange

Nixdorf Computer has announced the 8850 Model 85, an addition to its 8850 family of distributed data processing systems. The company says that the new model is intended to consolidate four middle-range 8850 models (35, 45, 55, and 65) into one field-upgradable system.

The 8850 Model 85 handles a maximum of 32 data terminals or workstations and a variety of peripherals. Users can also attach Nixdorf's 8810/25 and 8810/35 Compact and Desktop PCs as local workstations.

The new model uses Nixdorf's Dpx operating system.

A single cabinet consisting of CPU, tape drive, and disk drive costs \$48,000. A maximum configuration with four disk drives, two tape drives, CPU, and 32 terminals costs \$220,000. Contact Nixdorf Computer Corp., 80 Main St., North Reading, MA 01864; (617) 664-5781.

Reader Service Number 55

IBC adds Ensign CPU

Integrated Business Computers has announced an Intel 386-based Ensign microcomputer for heavy multitasker, multitasking applications. The Ensign 386:100 will run the Theos 386 operating system and support Xenix and multitasker MS-DOS. Currently, the system runs the Theos 286V OS.

A lower-end system for 16 users with 25M bytes of storage is priced at \$7995. A fully expanded system will provide 100 serial I/O ports, 24M bytes of main memory, 1M-byte serial and disk I/O buffer memory, four 68010 slave processors, a floppy disk drive, a tape cartridge drive, and three 280M-byte hard disk drives for \$71,395.

For more information, contact IBC, 21621 Nordhoff St., Chatsworth, CA 91311; (818) 882-9007.

Reader Service Number 56

VAXmate DBs link to VAX files

Network Innovations has announced a new version of its Multiplex software that links personal computer applications for Digital Equipment's VAXmate databases and data files on VAX superminicomputer systems.

The new version of Multiplex automatically handles all database inquiry, data reformatting, and data communications tasks, according to the company. It links the VAXmate to VAX/VMS and to Ultrix-32, DEC's version of Unix.

Multiplex is also available for IBM PCs and compatibles.

No prices given. For more information, contact Network Innovations, 20863 Stevens Creek Blvd., Cupertino, CA 95014; (408) 257-6800.

Reader Service Number 57

NEW LITERATURE

Strategic Alliances in the Pacific Rim and Indian Subcontinent. This impact report predicts that the US-Japan semiconductor trade agreement will drive US electronics manufacturers offshore. Electronic Trend Publications, 12930 Saratoga Ave., Ste. D1, Saratoga, CA 95070; (408) 996-7416; free.

New volume for scientists and engineers in the IBM PC Systems Handbook series. A 118-page handbook/catalog on instrumentation and data acquisition components and systems for IBM PC/XT/AT computers. All products were evaluated for functionality, overall quality, ease of use, cost effectiveness, mutual compatibility, and reliability. CyberResearch, Inc., 5 Science Park Ctr., PO Box 9565, New Haven, CT 06536; (203) 786-5151; \$12.95 for series subscribers.

The IBM XT Clone Buyer's Guide. Edwin Rutsch's book outlines the process of putting together an IBM clone. Also gives an overview of microcomputers and trends and advice on where to purchase an XT clone. Modular Information Systems, 431 Ashbury St., San Francisco, CA 94117; (415) 552-8648. \$9.95.

Black Box Corporation Catalog. Includes product listing, a technical reference section, and glossary. The company designs, manufactures, and distributes data communication products. Black Box Corp., PO Box 12800, Pittsburgh, PA 15241; (412) 746-5530; free.

Computer marketing program. A 52-page book, a six-page worksheet showing selections of the marketing components, and a worksheet for projecting each company's needs offer step-by-step information on how to plan a profit-based annual marketing program. omnistics, inc., 430 Airport Blvd., Salinas, CA 93905; (800) 345-7774; free.

"Future Directions in Computer Architecture and Software" proceedings. This workshop, held from May 5 to 7 at the Seabrook Conference Center, Seabrook, S.C., was attended by over 95 researchers in the field. Dr. C. Ronald Green, Electronics Div., Army Research Off., PO Box 12211, Research Triangle Park, NC 27709-2211; (919) 549-0641.

Recent Microsystem Announcements

MODEL AND FUNCTION	COMPANY	COMMENTS	R.S. NO.
Lightning f/i Internal modem	Anchor Automation 6913 Valjean Ave. Van Nuys, CA 91406 (818) 997-6493	A 2400 bps, full-card internal modem for IBM PCs and compatibles. Uses the MNP error correcting system. Comes with LYNC telecommunications software. Complies with V.22 bis and Bell 212A specifications. Cost: \$599.	80
T-120, T-160 Tape drives	CMS 3080 Airway Ave. Costa Mesa, CA 92626 (714) 549-9111	Add-on tape drives for the IBM XT 286 and Compaq 386, available in 20 and 60M-byte capacities. Both drives adhere to the QIC data format standard. The T-120 has four tracks, the T-160 nine. Cost: \$995 for the T-120, \$1595 for the T-160.	81
Codenet-3410, Computrol M-10F Fiber optic modems	Codenoll Technology 1086 N. Broadway Yonkers, NY 10701 (914) 965-6300	10M-bit/s fiber optic modems that operate with MAP. Designed to run with 802.4 token-bus controllers and computers using the VME or Multibus system buses. Cost: \$1750 each.	82
Chronos OS	Dynapro Systems 1000-1200 W. 73rd Ave. Vancouver, BC, Canada (604) 263-2638	A real-time, multitasking OS for the IBM PC-XT and PC-AT. Written in assembly language, tailored for the iAPX86 family of microprocessors. Cost: \$1995 US site license, \$2495 Canadian site license.	83
Forge Fortran optimizer	Pacific-Sierra Research 312 Main St., Suite 203 Placerville, CA 95667 (916) 621-1600	Unix-based Fortran optimization software for supercomputers, implemented on Apollo, MicroVAX II, and Sun workstations. Cost: \$99,000 for a 10-workstation corporate license; \$45,000 for a one-workstation license, with a \$5000 maintenance fee.	84
Smartmodem 1200C Internal modem	Hayes Microcomputer Products PO Box 105203 Atlanta, GA 30348 (404) 449-8791	An asynchronous, 1200-bps internal board modem for the IBM PC Convertible. Identical in function to the Smartmodem 1200 B. Features CCITT V.22 compatibility. Cost: \$439 (estimated).	85
KDS3 Expert system shell	KDS 934 Hunter Rd. Wilmette, IL 60091 (312) 251-2621	A frame-based expert system shell for the IBM PC, PC-XT, PC-AT, and compatibles. Offers all features of KDS2 and adds a blackboard and advanced math features. Development requires PC-DOS or MS-DOS, 512K-bytes of RAM, and a numeric coprocessor (8087, 80287, or 80387). Written in assembly language. Cost: \$1495 for a one-workstation development license.	86
RISP-II Signal processor	Matsushita Electric One Panasonic Way Secaucus, NJ 07094 (201) 348-7320	A real-time image signal processor. Reputedly operates at a 10-ns speed. Integrates 20,600 elements on each LSI chip. Reprogrammable. Cost: 250,000 yen for one sample.	87
Mitrol-Micro 4GL	Mitrol 800 W. Cummings Park Woburn, MA 01801 (617) 933-9545	A fourth-generation language for the IBM AT/370 and compatibles. Applications will also run on an IBM mainframe operating Mitrol under VM/CMS, OS/MVS, or OS/MVS/XA. Cost: \$3000 to \$25,000.	88
poly-Star Software	Polygon Assoc. 1024 Executive Pkwy. St. Louis, MO 63141 (314) 576-7709	A terminal emulation and file transfer program for communication among IBM PC, VAX, and PDP system users. Permits remote control of the PC from the host. Incorporates features of poly-Com/220 and 240 software. Cost: From \$200.	89

ADVANCE PROGRAM

THE THIRD IEEE ARTIFICIAL INTELLIGENCE APPLICATIONS

Conference and Exposition

Orlando Hyatt Regency
Kissimmee, Florida
February 23-27, 1987

Sponsored By
 The Computer Society of the IEEE



THE CONFERENCE

The Third IEEE Artificial Intelligence Applications Conference and Exhibition will be held February 23-27, 1987, in Kissimmee, Florida. The conference concentrates on the application of artificial intelligence techniques to real-world problems.

TECHNICAL SESSIONS

The technical sessions feature the following topics:

- | | | |
|----------------------------|------------------------------|------------------------------|
| * Knowledge Acquisition | * Question Answering | * Manufacturing |
| * Knowledge Representation | * Explanation-Based Learning | * AI & Real Time Programming |
| * Diagnosis | * Robotics and Perception | * Uncertainty |
| * Case Studies | * Search | * Software & Tools |
| * Default Reasoning | * Design and Planning | |

TUTORIALS

There will be eight half-day tutorials on the following:

- * Managing Knowledge System Development, Avron Barr
- * Programming in the Lisp Machine Environment, Sue Green
- * Analyzing Expert System Building Tools, Paul Harmon
- * Logic Programming, Expert Systems and Databases, Steve Hardy
- * AI and Computer Integrated Manufacturing, Arvind Sathi
- * Commercial Applications of Natural Language Processing, Tim Johnson
- * AI Programming on Parallel Machines, Joe Brandenburg
- * Intelligent Interfaces, Marilyn Stelzner

EXHIBITS

In addition to the informative conference sessions, our exhibit floor will offer you the opportunity to view and meet with...

Over 50 different exhibits by the leaders in AI; the leading publishers in the computer book and periodical market; the systems developers who are designing the Human-Computer interface of the future; the programmers that are creating natural languages that enable computers to perform knowledge acquisition and reasoning, planning and problem solving; the manufacturers that are building the systems of tomorrow, that you want to see and use today!

FOR FURTHER INFORMATION

For Exhibits Contact: Mark Bakke, Trade Associates, 12250 Rockville Pike, Suite 200, Rockville, Maryland 20852, (301) 468-3210.

For Conference Information and a complete brochure contact: CAIA Conference, Computer Society of the IEEE, 1730 Massachusetts Ave., N.W., Washington, DC, 20036-1903, (202) 371-0101.



THE COMPUTER SOCIETY
OF THE IEEE



THE INSTITUTE OF ELECTRICAL
AND ELECTRONICS ENGINEERS INC

IEEE

The Third IEEE Artificial Intelligence Applications Conference and Exhibition



ADVANCE CONFERENCE REGISTRATION FORM February 23-27, 1987; Hyatt Orlando, Kissimmee, Florida

Complete and return this form with your check (in U.S. dollars) payable to CAIA or charge it to your MasterCard, Visa, American Express or Choice using the form below.

Send to: CAIA

Computer Society of the IEEE, 1730 Massachusetts Avenue, N.W., Washington, DC
20036-1903, Phone: (202) 371-0101, TWX7108250437IEEECOMPSO

Circle Appropriate Fee:

	Advance Registration (Prior to 2/6/87)			Late Registration (After 2/6/87)		
	Member	Non-Member	Student	Member	Non-Member	Student
1 Tutorial	\$110	\$140	\$110	\$140	\$175	\$140
2 Tutorials	\$220	\$280	\$220	\$280	\$350	\$280
3 Tutorials	\$330	\$420	\$330	\$420	\$525	\$420
4 Tutorials	\$440	\$560	\$440	\$560	\$700	\$560
Conference Only	\$175	\$220	\$ 50	\$200	\$250	\$ 50
Conference & 1 Tutorial	\$285	\$360	\$160	\$340	\$425	\$190
Conference & 2 Tutorials	\$395	\$500	\$270	\$480	\$600	\$330
Conference & 3 Tutorials	\$505	\$640	\$380	\$620	\$775	\$470
Conference & 4 Tutorials	\$615	\$780	\$490	\$760	\$950	\$610

TUTORIALS: Please check appropriate tutorial. (only two per day)

Monday, 2/23: 8:30-12:00

- ___ 1. Managing Knowledge System Development
___ 2. Programming in the Lisp Machine Environment

Tuesday, 2/24: 8:30-12:00

- ___ 5. AI and Computer Integrated Manufacturing
___ 6. Commercial Applications of Natural Language Processing

Monday, 2/23: 1:30-5:00

- ___ 3. Analyzing Expert Systems Building Tools
___ 4. Logic Programming, Expert Systems and Databases

Tuesday, 2/24: 1:30-5:00

- ___ 7. AI Programming on Parallel Machines
___ 8. Intelligent Interfaces

Registration includes 1 year subscription to IEEE Expert Magazine.

Name _____

Company Name _____

Address _____

City _____ State _____ Zip _____

Phone _____

IEEE Member Number _____

Total Fees enclosed: _____

___ Check Enclosed ___ MasterCard ___ VISA ___ American Express ___ Choice

Card No. _____ Exp. Date _____

Signature _____

NOTE:

- * Student fees are for full time student members of the IEEE and do not include proceedings or the subscription.
- * Requests for refunds must be received in writing no later than 2/20/87 and are subject to a \$15 cancellation fee.
- * Tutorial registration includes luncheon(s), if attending a morning and a afternoon tutorial on the same day, coffee breaks and tutorial notes.
- * Conference registration includes one copy of the proceedings, reception, subscription to Expert, and coffee breaks.

HOTEL RESERVATION FORM HYATT ORLANDO WELCOMES COMPUTER SOCIETY OF THE IEEE THIRD CONFERENCE ON ARTIFICIAL INTELLIGENCE APPLICATIONS FEBRUARY 23-27, 1987

DAILY ROOM RATES \$80.00 SINGLE; \$90.00 DOUBLE

\$90.00 SINGLE; \$90.00 QUAD

All requests must be received by the hotel no later than 1/24/87

Check-out time is 4:00 P.M.

ALL RATES ARE SUBJECT TO 8% STATE TAX

SPECIAL ROOM REQUESTS ARE ON A SPACE AVAILABLE BASIS AND CANNOT BE CONFIRMED OR GUARANTEED.

CHILDREN UNDER 18 YEARS OF AGE FREE IN ADDITION TO DOUBLE OCCUPANCY A WRITTEN CONFIRMATION WILL BE SENT TO YOU

Check-out time is 12:00 Noon. Baggage should be checked with the bell captain if departure is later than 12:00 Noon.

(Hotel telephone 305-396-1234)

THIS IS A RESERVATION REQUEST AND MUST BE ACCOMPANIED BY A ONE NIGHTS ROOM DEPOSIT OR CREDIT CARD GUARANTEE.

TYPE _____ EXP _____

ARRIVAL DATE _____ DEPARTURE DATE _____

NUMBER IN PARTY _____ ADULTS _____ CHILDREN _____

SPECIAL REQUESTS _____

NAME _____

FIRM _____

ADDRESS _____

CITY _____ STATE _____ ZIP _____

TELEPHONE () _____

Eastern Airlines is offering conference attendees 50% off regular coach fares. Please call World Wide Travel at 800-222-2920 or 202-659-1121 for further information.

BOOK REVIEWS

Microcomputers and Microprocessors

John Uffenbeck (Prentice-Hall, Englewood Cliffs, N.J., 1985, 670 pp., \$31.95)

My first reaction on seeing this book was, "Here's a Johnny-come-lately!" A book on the 8080 in a world going the 32-bit way? However, there are some saving graces, and the author does justify his efforts. The book should be of interest to anyone using the 8080, 8050, or Z-80. However, with the thorough coverage of memory technologies, serial communications, magnetic disk concepts, analog interfacing, and microcomputer troubleshooting, the book is more a microcomputer text than it is a book about these three processors.

The book is written in a casual, yet easily understood, style. It is suitable for undergraduates or first-year graduate students. The practicing engineer might also find it a useful book for his or her library.

The author places particular emphasis on visualizing the microprocessor as part of a CPU module that is, in turn, interfaced by the memory and I/O. He pays close attention to concepts of bus buffering such as noise immunity, DC loading, and reflections.

There is an entire chapter devoted to the 8086, which can be seen as an extension of the 8-bit processors discussed earlier in the book. The point-wise chapter summaries are a good idea; so is the section on troubleshooting techniques.

Some highlights:

The author stresses that microprocessor programming is best learned in context. To this end he discusses 14 detailed programs in Chapter 3, illustrating new instructions and addressing modes. Chapter 3 also introduces CP/M, an industry-wide operating system for 8080/85 and Z-80 microcomputer systems.

Chapter 5 shows how to interface RAM and ROM memory to the CPU module developed earlier. Timing requirements imposed by the microprocessor are defined and the memory designs analyzed to verify that these constraints have been met.

Chapter 6 concludes "construction" of the microcomputer with the addition of the I/O module. The microprocessor

IN and OUT instructions are used to identify the I/O-and memory-mapped hardware requirements. The three common methods of controlling I/O devices—polling, interrupts, and DMA—are contrasted and response-time and transfer rates are calculated for each.

Chapters 7 and 8 deal with the 8080/85 and Z-80 family respectively because the special-purpose support devices discussed are sufficiently different to warrant separate chapters.

An interesting feature of the book is the 40 or more data sheets distributed throughout the text as appropriate. I agree with the author that we must all learn to read the literature distributed by the semiconductor manufacturers.

Going from Basic to C

Robert J. Traister (Prentice-Hall, Englewood Cliffs, N.J., 162 pp., \$17.95)

In the Preface the author states that the book is written especially for the Basic language programmer who is planning to make the switch to C language. Since this description fit my situation, I felt eager to investigate this book. The author's goals were to direct the reader: (1) to begin writing simple C programs, (2) to fully understand the relationship between Basic statements and C functions, and (3) to build adequate knowledge to effectively use C instruction manuals. The author has done a superb job in attaining his goals!

Some highlights:

After the usual historical notes and a description of the C compiler used for the examples, the author describes simple functions for reading the keyboard and writing to the screen. Programs are first shown in Basic and then shown in C. After looking at these C examples, I began to feel at ease with the C syntax.

The section about loops was well done. Since we Basic programmers tend to use GOTO statements more often than programmers in other languages, there is a "commercial" about not using

One failing worth mentioning is the lack of discussion on new architectures such as RISC machines, and perhaps the author could have mentioned, at least in passing, multimicroprocessors.

On the whole, I think the author has succeeded in his two main goals:

(1) to provide a practical microcomputer text with an emphasis on the programmable peripheral controller support chips, and

(2) to use the Intel 8080, 8085, and the Zilog Z-80 microprocessors as realistic "vehicles" to demonstrate these concepts.

Chandan Sen
Research Triangle Institute

GOTOs and examples of how to avoid them.

In order to emphasize traps that Basic programmers might fall into, several examples of often-made errors are given. These wrong examples are clearly marked so that even if the reader just scans the listings, he or she will not be misled.

Chapter 7 contains 15 examples with errors. The first six chapters must have been presented well, since I was able to identify 12 of these 15 errors.

I don't find C programs to be so ominous after reading the chapter called, "Characters, Conversions and Confusion." Data types and the ease of C conversion methods are much clearer.

The chapter on mimicking Basic statements could have been longer. In fact it would have been useful to have had an appendix of Basic statements vs. C functional equivalents.

My suggestion to a would-be C programmer who has a background in Basic is to read this book first and then get into one of the more complete C references. I wouldn't want to learn a foreign language unless the explanatory text were written in English. This book makes "speakers" of Basic more conversant in C.

Gordon W. Flemming
Jet Propulsion Laboratory

CALENDAR

January 1987

20th Hawaii International Conference on System Sciences, January 6-9, Kailua-Kona, Hawaii. Contact Ralph H. Sprague, University of Hawaii, College of Business Administration, R-303, Honolulu, HI 96822; (808) 948-7430.

Image Pattern Recognition: Algorithm Implementations, Techniques, Technology (a Critical Review Conference) (SPIE), January 11-16, Los Angeles. Contact Joseph Yaver, SPIE, PO Box 10, Bellingham, WA 98227; (206) 676-3290.

Image Understanding and the Man-Machine Interface (SPIE), January 11-16, Los Angeles. Contact Joseph Yaver, SPIE, PO Box 10, Bellingham, WA 98227; (206) 676-3290.

Optical and Digital Pattern Recognition (SPIE), January 11-16, Los Angeles. Contact Joseph Yaver, SPIE, PO Box 10, Bellingham, WA 98227; (206) 676-3290.

IC-87, Symposium on Intelligent Control, January 18-21, Philadelphia. Contact A. Meyster, Drexel University, Dept. of Electrical and Computer Engineering, College of Engineering, Philadelphia, PA 19104; (215) 895-2220.

Fourth International Symposium on Modeling and Simulation Methodology: Intelligent Environments, Goal-Directed Models (SCS), January 19-23, Tucson, Arizona. Contact Paul A. Baltes, Office of Special Professional Education, University of Arizona, Havville Bldg., Box 9, Tucson, AZ 85721; (602) 621-3054.

Annual IEEE Design Automation Workshop, January 21-23, Apache Junction, Arizona. Contact Jim Armstrong, Electrical Engineering Dept., Virginia Tech, Blacksburg, VA 24061; (703) 961-4723 or (703) 961-7078.

1987 Winter Usenix Technical Conference, January 21-23, Washington, DC. Contact Usenix Conference Office, PO Box 385, Sunset Beach, CA 90742; (213) 592-1381 or (213) 592-3243.

IEEE Workshop on Instrumentation for Distributed Computing Systems, January 22-23, Sanibel Island, Florida. Contact Andre van Tilborg, Computer Science Dept., Carnegie Mellon University, Pittsburgh, PA 15213; (412) 268-3801.

February 1987

Third International Conference on Data Engineering, February 2-6, Los Angeles. Contact Gio Wiederhold, Dept. of Computer Science, Stanford University, Stanford, CA 94305; (415) 497-0685 or Benjamin Wah, Coordinated Science Lab, University of Illinois, Urbana, IL 61801; (217) 333-5216.

CSC-87, 1987 ACM Computer Science Conference, February 17-19, St. Louis. Contact Arlan DeKock, University of Missouri-Rolla, Computer Science Dept., Rolla, MO 65401; (314) 341-4491.

18th Annual SIGSCS Technical Symposium (ACM), February 19-20, St. Louis. Contact Dan C. St. Clair, UMR Graduate Engineering Center, University of Missouri-Rolla, 8001 Natural Bridge Rd., St. Louis, MO 63121-4499; (314) 553-5440.

Compeon Spring 87, February 23-26, San Francisco. Contact Glen G. Langdon, Jr., IBM Dept. K54-802, 450 Harry Rd., San Jose, CA 95120-6099; (408) 927-1818.

Third IEEE Conference on Artificial Intelligence Applications, February 23-27, Orlando, Florida. Contact Jan Aikins, Aion Corp., 101 University Ave., fourth floor, Palo Alto, CA 94301; (415) 328-9595.

March 1987

20th Annual Simulation Symposium (ACM, IMACS, SCS), March 11-13, Tampa, Florida. Contact James Gantt, 5986 Dana Dr., Norcross, GA 30093; (404) 894-3107.

Sixth Symposium on Reliability in Distributed Software and Database Systems (NASA), March 17-19, Williamsburg, Virginia. Contact Edwin C. Foudrist, NASA, Langley Research Center, Information Systems Division, MS 469, Hampton, VA 23665; (804) 865-3535.

1987 IEEE VLSI Test Workshop, March 24-25, Atlantic City, New Jersey. Contact Wesley E. Radcliffe, IBM East Fishkill, Dept. 277, Bldg. 321-5E1, Hopewell Junction, NY 12533; (914) 894-4346.

Southeast (IEEE), March 24-26, Atlanta. Contact Electronic Conventions Management, 8110 Airport Blvd., Los Angeles, CA 90045; (213) 772-2965.

IEEE Infocom 87, Sixth Annual Joint Conference of the IEEE Computer and Communications Societies: Global Networks—Concept to Realization, March 30-April 2, San Francisco. Contact IEEE Infocom 87, 1730 Massachusetts Ave. NW, Washington, DC 20036-1903; (202) 371-0101; telex 7108250437 IEEECOMPSO.

Ninth IEEE International Conference on Software Engineering (ACM), March 30-April 2, Monterey, California. Contact William Riddle, Software Design and Analysis, 1760 Bear Mountain Dr., Boulder, CO 80303; (303) 499-4782.

Fourth International Symposium on Optical and Optoelectronic Applied Science and Engineering (SPIE), March 30-April 3, The Hague, The Netherlands. Contact Joseph Yaver, SPIE, PO Box 10, Bellingham, WA 98227-0010; (206) 676-3290.

1987 IEEE International Conference on Robotics and Automation, March 30-April 3, Raleigh, North Carolina. Contact Harry Hayman, 738 Whitaker Terrace, Silver Spring, MD 20901; (301) 434-1990.

April 1987

Fourth International Workshop on Software Specification and Design (AFCT, Agence de l'Informatique, Alvey Directorate, LCRST-Japan), April 3-4, Monterey, California. Contact David Marica, Digital Equipment Corp., Intelligent Systems Tech Group, 77 Reed Rd., MS HL02-3/CID; Hudson, MA 01749.

IEEE Computer Society Symposium on Office Automation: Integration, Interconnection, and Use of Personal Computers (NBS), April 27-29, Gaithersburg, Maryland. Contact Vincent Lum, Dept. of Computer

Conferences that the Computer Society participates in or sponsors are indicated by the IEEE Computer Society logo; other conferences of interest to our readers are also included. For inclusion in Calendar or Call for Papers, submit information six weeks before the month of publication (e.g., for the March 1987 issue, send information for receipt by January 15, 1987) to COMPUTER, 10662 Los Vaqueros Circle, Los Alamitos, CA 90720.

Science, Naval Postgrad School, Monterey, CA; (408) 646-2449.

IEEE Symposium on Security and Privacy, April 27-29, Oakland, California. Contact Virgil Gligor, University of Maryland, Dept. of Electrical Engineering, College Park, MD 20742; (301) 454-8846.

May 1987

Eighth IEEE Symposium on Mass Storage Systems, May 11-14, Tucson, Arizona. Contact Patrick Savage, Shell Development Co., PO Box 481, Houston, TX 77001; (713) 663-2384.

Third International Symposium on VLSI Technology, Systems, and Applications, May 13-15, Taipei, Taiwan. Contact Ben M.Y. Hsiao, IBM Corp., Dept. D18, Bldg. 707, PO Box 390, Poughkeepsie, NY 12602 or Morris Chang, TTRI, No. 1, Jen Ai Rd., Sec 2, Taipei, Taiwan.

Eighth Symposium on Computer Arithmetic, May 18-21, Como, Italy. Contact Mary Jane Irwin, Dept. of Computer Science, 333 Whitmore Laboratory, Pennsylvania State University, University Park, PA 16802 or Luigi Dadda, Dept. of Electronics, Piazza Leonardo da Vinci 32, Politecnico di Milano, I-20133 Milan, Italy.

1987 International Symposium on Multiple-Valued Logic, May 26-28, Boston. Contact Dan Simovic, Dept. of Mathematics and Computer Science, University of Massachusetts, Boston, MA 02125; (617) 929-7966.

June 1987

IFIP Workshop on CAD Engines, June 1-2, Kakaishinlou, Kainan, Japan. Contact Tatsuo Ohtsuki, Waseda University, Dept. of Elec. and Comm., School of Science and Engineering, 3-4-2 Okubo, Shinjuku, Tokyo 160, Japan; phone 81 (03) 209-3211.

14th International Symposium on Computer Architecture, June 2-5, Pittsburgh. Contact Zary Segall, Computer Science Dept., Carnegie Mellon University, Pittsburgh, PA 15213; (412) 268-3736.

ICCV-87, First International Conference on Computer Vision, June 8-11, London. Contact Azriel Rosenfeld, University of Maryland, Center for Automation Research, College Park, MD 20742; (301) 454-4526.

IEEE Workshop on Software Technology Transfer, June 11-13, Santa Fe, New Mexico. Contact Charles Richter, MCC, Software Tech. Prog., 9430 Research Blvd., Austin, TX 78759.

CALL FOR PAPERS

Computer: Articles that cover the state of the art and important new developments in computer science, technology, and applications are sought. Submit materials to Bruce Shriver, IBM T.J. Watson Research Center, Route 134, PO Box 218 (HO-B04A), Yorktown Heights, NY 10598; (914) 945-1664. His electronic mail addresses are as follows: For Comppmail +, b.shriver; for CSnet, shriver@ibm.com; and for Vnet, shriver at ykvtm.

CG International 87, Fifth International Conference on Computer Graphics in Japan (CCS): May 25-28, 1987, Karuizawa, Nagano Prefecture, Japan. Submit five copies of a paper of between 5000 and 10,000 words (include title, abstract of up to 150 words, a maximum of 10 key words and phrases, short biography, photo, and commitment to present paper) by December 15, 1986, to Toshiyuki L. Kunii, c/o Dept. of Information Science, Faculty of Science, the University of Tokyo, Hongo, Tokyo 113, Japan.

ICCV-87, First International Conference on Computer Vision: June 8-11, 1987, London. Send four copies of complete paper by December 15, 1986, to Azriel Rosenfeld, Center for Automation Research, University of Maryland, College Park, MD 20742.

ICDCS-7, Seventh International Conference on Distributed Computing Systems: September 21-25, 1987, Berlin. Submit five copies of paper (20 pages maximum) by January 1, 1987, to G. Le Lann, INRIA, Rocquencourt, BP 105, 78153 Le Chesnay Cedex, France; phone 33 (1) 39-63-5364 (authors in Europe or Africa) or K.H. Kim, Computer Engineering Program, Dept. of Electrical Engineering, Rm. 544B, University of California, Irvine, CA 92717; (714) 856-4821 (authors in the Americas, Asia, or Australia). Proposals for one-day tutorials are also sought. Submit them by the same date to Earl Swartzlander Jr., TRW Defense Systems Gr., 1 Space Park 02/2791, Redondo Beach, CA 90278 or L. Pouzin, CNET Centre Paris A, 38-40 Rue de Gen. Leclerc, 9231 Issy les Moulineaux, France.

Third International Symposium on VLSI Technology, Systems, and Applications: May 13-15, 1987, Taipei, Taiwan. Submit 30 copies of a 300- to 600-word summary (include name, affiliation, address, and telephone number), and a 35- to 50-word abstract by January 1, 1987, to Ben M.Y. Hsiao, IBM Corp., Dept. D18, Bldg. 707, PO Box 390, Poughkeepsie, NY 12602.

International Workshop on Petri Nets and Performance Models: August 24-26, 1987, Madison, Wisconsin. Submit six copies of a

complete paper (20 pages maximum) by January 2, 1987, to Michael K. Molloy, 4212 Wean Hall, Dept. of Computer Science, Carnegie Mellon University, Pittsburgh, PA 15213.

CSM-87, Conference on Software Maintenance (AWC, DPMA, NBS, SMA): September 21-24, 1987, Austin, Texas. Submit five copies of papers (1000 to 5000 words; paper should be accompanied by a 250-word abstract) and five copies of proposals for panel sessions (include a list of four to five potential panelists) by January 6, 1987, to Wilma M. Osborne, National Bureau of Standards, Bldg. 266, Rm. B266, Gaithersburg, MD 20899; (301) 912-3545.

1987 International Conference on Parallel Processing: August 17-21, 1987, St. Charles, Illinois. Regular-length and short papers are sought. For regular-length papers, submit four copies each of the full text and a 100-word abstract. For short papers, submit four copies each of a 1000-word summary and a 100-word abstract. Submit materials in both categories by January 15, 1987, to Sartaj K. Sahni, Dept. of Computer Science, 136 Lind Hall, University of Minnesota, Minneapolis, MN 55455.

Sixth ACM SIGACT-SIGOPS Symposium on Distributed Computing: August 10-12, 1987, Vancouver, British Columbia, Canada. Submit a detailed abstract (10 pages maximum) by January 30, 1987, to Fred B. Schneider, Dept. of Computer Science, Upson Hall, Cornell University, Ithaca, NY 14853.

Compasac 87: October 5-9, 1987, Tokyo. Papers and proposals for panel discussions are sought. Submit five copies of the full-length paper (1000 to 5000 words; include a 150-word abstract). Submit five copies of proposals for panel discussions (include names of proposed panelists and a 150-word scope statement). Materials in both categories must be received by February 1, 1987. Send them to Toshiyuki L. Kunii, c/o Business Center for Academic Societies Japan, Yamazaki Bldg. 4F, 2-40-14, Hongo, Bunkyo-Ku, Tokyo 113, Japan; phone 81 (3) 817-5831 (authors in Asia, Australia, and New Zealand) or Albert K. Hawkes, Sargent & Lundy, Engineering Consultants, 55 E. Monroe, Chicago, IL 60603; (312) 269-3640 (authors in other areas).

IEEE Transactions on Computers: Papers are sought for a special issue on supercomputing. Submit six copies of papers by February 1, 1987, to H.C. Torng, School of Electrical Engineering, Cornell University, Ithaca, NY 14853-5401; (607) 255-5191. (Guidelines for submitting manuscripts appear on the back cover of every issue of IEEE Transactions on Computers.)

CAREER OPPORTUNITIES

RATES: \$12 per line, \$120 minimum charge (up to ten lines). Average six typeset words per line, nine lines per column inch. Add \$10 for box number. Send copy at least six weeks prior to month of publication to: **Carole L. Porter, Classified Advertising, COMPUTER Magazine, 10562 Los Vaqueros Circle, Los Alamitos, CA 90702.**

To allow sufficient time for distribution of the magazine, schedule your job application deadlines at least four weeks after issue date (e.g., deadline of Nov. 28 to send applications responding to ad appearing in Nov. issue).

Note: IEEE Computer Society member ads for positions wanted \$10 up to 10 lines, \$1 per line thereafter. Situations wanted ads run free for unemployed society members. Include IEEE member number in either case.

In order to conform to the Age Discrimination in Employment Act and to discourage age discrimination, **IEEE COMPUTER** may reject any advertisement containing any of these phrases or similar ones: "... recent college grads...", "... 1-4 years maximum experience...", "... up to 5 years experience...", "... 10 years maximum experience..." or "... **COMPUTER** reserves the right to append to any advertisement, without specific notice to the advertiser, "Experience ranges are suggested minimum requirements, not maximums." **IEEE COMPUTER** assumes that, since advertisers have been notified of this policy in advance, they agree that any experience requirements, whether stated as ranges or otherwise, will be construed by the reader as minimum requirements only.

SYRACUSE UNIVERSITY Dean School of Computer and Information Science

Syracuse University invites nominations and applications for the position of Dean of the School of Computer and Information Science. The Dean will lead an expanding faculty, currently 20 in number, active in a broad program of research with major concentrations in computational logic, programming language theory, and coding and information theory. There are also about 30 affiliated faculty in other units of the University. The School has approximately 200 undergraduate, 100 master's degree, and 40 Ph.D. students on the main campus, and also offers graduate programs at three extension sites. The School will move to a new 130,000-square-foot building in September, 1988. Other units, including the Center for Advanced Technology in Computer Applications and Software Engineering (the CASE Center) and the twenty-member computer engineering faculty, will also move to the new building. The School is associated with the research programs of the CASE Center and the Northeast Artificial Intelligence Consortium, and is one of the units involved in the founding of a center for the study of parallel architectures. Candidates should have an established reputation in research and some experience in academic administration. They should be committed to the advancement of computer science and to excellence in scholarship and teaching. The position requires the vision to plan and the leadership ability to direct the School's recruitment efforts, expansion into new research areas, and procurement of external funds.

Syracuse University, a private institution, has major programs of research, graduate studies, and undergraduate education. There are 11,800 undergraduate and 4,400 graduate students on the main campus.

Application and Nomination Procedures. Applicants should send resumes, names of references, and selected publications to: Dean Search Committee, School of Computer and Information Science, 313 Link Hall, Syracuse University, Syracuse, New York 13244-1240, by January 19, 1987. Letters of nomination should be sent by the same date to the above address. The appointment is expected to be effective no later than September 1, 1987. Syracuse University is an equal opportunity, affirmative action employer.

UNIVERSITY OF CALIFORNIA AT DAVIS

The Computer Science Division of the University of California at Davis invites applications from highly qualified individuals for several tenure-track positions. The Division is especially interested in candidates whose research is in the fields of computer architecture, networks and distributed systems, artificial intelligence, programming languages/compilers, and VLSI design. Outstanding candidates in other areas, except numerical analysis, will also be considered. Rank and salary will be commensurate with qualifications. All applicants should have a Ph.D. in Computer Science or Computer Engineering. New or recent graduates should show evidence of outstanding research potential, while more senior applicants are expected to have a significant and active research record. The Division of Computer Science was established three years ago as an independent unit within the Department of Electrical and Computer Engineering. Its interests cover the mainstream areas of computer science, including artificial intelligence, computer systems design, database systems, computer graphics, fault tolerant computers, networking, numerical analysis, programming languages, operating systems, performance evaluation, robotics, software engineering and theoretical computer science.

The Division is in a period of building and growth with the aim of becoming one of the leading computer science departments in the nation. Its expanding departmental research facilities include a VAX 8600, VAX 11-785, several VAX 11-750s, a Ridge 32, several Sun, Microvax II and Apollo work stations, and a number of special purpose systems all linked through ETHERNET and serial communications and networks.

General purpose campus facilities are available, and the supercomputer resources of the Lawrence Livermore National Laboratories are also accessible. Current research strengths and facilities exist in the areas of image processing, computer graphics, computer systems research, microprocessors and robotics. UC Davis is the third largest of the University of California campuses. It shares with the other campuses of the UC system a strong commitment to leadership in education and research. Davis is close to the major areas of computer science and electronics activity in northern California and thus is in a strategic position to

play a key role in future developments of the discipline.

The town of Davis has a population of approximately 50,000. Located eleven miles from Sacramento, it is situated within a short driving distance from the major cultural centers of the San Francisco area as well as the outstanding recreational facilities of the Sierra Nevada Mountains. The town has earned an international reputation for its energy conservation and progressive growth policies. Because of its location, its climate, and its sophisticated small town character, Davis is considered by many an ideal living environment.

Interested applicants should send a curriculum vitae with the names of at least three professional references to Professor Richard F. Walters, Division of Computer Science, or to Professor Lou Hakimi, Department of Electrical and Computer Engineering, University of California, Davis, CA 95616.

The University of California is an equal opportunity/affirmative action employer.

UNIVERSITY OF MARYLAND Information Systems

The Information Systems department has a tenure-track position available at the Associate or Assistant Professor level. Applicants with an interest in both research and teaching are being sought. Attractive salaries are offered.

Information Systems is an active, growing department at the University of Maryland. The graduate program attracts quality students for M.S. and Ph.D. degrees in Information Systems. Current research in our Database Systems Research Center includes topics in database systems optimization, database design, office information systems, knowledge based systems, and systems analysis and design. For advantageous consideration, applications should be received by February 1, 1987. Candidates should send complete resumes and names of three references to: Prof. Alan R. Hevner, Search Committee Chairperson, Information Systems, College of Business and Management, University of Maryland, College Park, MD 20742, (301) 454-3260. The University of Maryland is an Equal Opportunity, Affirmative Action Employer. Women and minorities are encouraged to apply.

Systems Analyst

Design and develop the specifications and lead the implementation of information management and training management systems. Modify programs to meet client special requirements. Debug, troubleshoot and provide client support and programmer training. Knowledge of or training with Data Base and File Management systems and information processing. Familiarity with microcomputer architecture software and systems. Knowledge of image processing technology and computer graphics.

M.S. or B.S. and two years experience in Computer Science. Salary \$2730 per month. Job site: Walnut Creek, CA. Please send this ad and a resume to Job #FC 8739, P.O. Box 9560, Sacramento, CA 95823-0560 not later than December 15, 1986.

GEORGE WASHINGTON UNIVERSITY
Tenure-Track Faculty Positions

The Department of Electrical Engineering and Computer Science invites applications for tenure-track faculty positions. We particularly seek persons active in the areas of Computer Science and Communications. Candidates should have an earned doctorate and research experience, with an interest in both teaching and research. Ability to attract research grants and contracts for support of faculty and student assistants is valued. The George Washington University is located in the center of Washington, DC. This metropolitan area sustains the second largest concentration of research and development activity in the United States which creates a continuing demand for rigorously trained engineers as well as broad research opportunities. Our department is the largest department in the School of Engineering and Applied Science with 33 full-time faculty, and a large graduate and undergraduate student body. The department has an annual research budget of close to \$1 million. Send Curriculum Vitae, list of publications and references, to:

Professor Roger H. Lang, Chairman
Department of Electrical Engineering and Computer Science
School of Engineering & Applied Science
THE GEORGE WASHINGTON UNIVERSITY
Washington, DC 20052
The University is an affirmative action/equal opportunity employer.

WASHINGTON UNIVERSITY, ST. LOUIS
Regular Faculty Positions in
Computer Science

The Department of Computer Science is expanding its research program and invites applications for several regular (tenure-track) faculty positions at the Assistant, Associate and Full Professor levels. Applicants should hold the Ph.D. or D.Sc. degree in Computer Science and have a strong commitment to and record of accomplishment in research.

The Department of Computer Science has a well-respected undergraduate program and a small, but growing graduate program. Research activities have flourished in recent years and have three foci: concurrent systems, communications systems, and intelligent computer systems. Support for these research programs is from NSF, NIH, ONR, DMR and a large number of industrial sponsors.

Current departmental research in concurrent systems includes the formal foundations of concurrent computation, methodologies for the design of distributed systems, system interconnection networks and parallel architectures and algorithms. The department is also engaged in research on the design of high-performance packet-switching systems capable of a variety of applications (including voice, data and video). Of particular interest are candidates with backgrounds in communications systems architecture, communications software, or performance analysis. In the area of intelligent computer systems, present research projects include computer vision, image processing, visual programming, speech recognition, expert systems and deductive databases. Candidates with backgrounds in any of these areas are encouraged to apply.

Candidates may reply to:

Dr. Jerome R. Cox, Jr., Chairman
Department of Computer Science
Washington University
Campus Box 1045

St. Louis, Missouri 63130

Applications received by February 1, 1987. Washington University is an equal opportunity/affirmative action employer.

CITY COLLEGE OF NEW YORK
Faculty Positions

Applications are invited for anticipated tenure-track faculty positions in the Department of Electrical Engineering. Applicants should possess a Ph.D. and have a strong commitment to teaching and research. Candidates should have experience at least in one of the following three areas: computer engineering, signals and systems (power, control, communications, signal and image processing, robotics) and Electrophysics (microwaves, optics and lasers). An outstanding research reputation, with an ability to attract external funding, for senior positions, or a demonstrated research potential, for junior positions, is desirable. Rank and salary are open. Application, including recent publications and research interest, names of three professional references should be sent to: Faculty Search Committee, Department of Electrical Engineering, The City College of CUNY, Contract Ave at 138th Street, New York, 10031. (212) 690-4248. Applications will be received until all positions are filled. An EO/AA Employer.

**CARLETON UNIVERSITY—SCHOOL OF
COMPUTER SCIENCE**

Applications are invited for a tenure track position at the rank of Assistant or Associate Professor in the School of Computer Science starting July 1, 1987. Teaching duties will include both undergraduate and graduate instruction in Computer Science. Outstanding candidates with a Ph.D. in any area of computer science and engineering will be considered, with particular preference given to those in software engineering, artificial intelligence, database systems, distributed computing, operating systems, computer architecture and computer graphics. Prospective faculty can look forward to a dynamic research environment in which collaboration with industry is encouraged. The undergraduate honours programs and graduate programs have restricted enrollment in order to maintain a high quality research and teaching environment.

Salary commensurate with qualifications and experience. Send curriculum vitae and names of three references to Professor John Pugh, Director, School of Computer Science, Carleton University, Ottawa, Ontario, K1S 5B6, Canada. Open to both men and women. In accordance with Canadian Immigration requirements, this advertisement is directed to Canadian citizens and permanent residents. Hiring is subject to budgetary approval.

LOUISIANA STATE UNIVERSITY
Faculty Openings

The Department of Electrical and Computer Engineering at LSU invites applications for tenure-track and visiting faculty positions available August 1987 in all areas of computer engineering and special purpose architectures. A Ph.D. or equivalent and potential for excellence in teaching and research are necessary. Rank is open. Salary is competitive and commensurate with qualifications and experience. Release time and resources are provided in order to enhance the development of a quality research program. Opportunities for summer support are available. Send resume, names of three references and a statement of teaching and research interests to: Alan H. Marshak, Chairman, Electrical and Computer Engineering, Louisiana State University, Baton Rouge, LA 70803-5901. LSU is an Equal Opportunity Employer.

SENIOR SYSTEMS ANALYST

Will provide ongoing monitoring of Distribution Center Management System (DCMS) performance; change/impact analysis regarding user change requests; manage test environments; trouble shoot system related problems; participate in control of future DCMS releases; subsequent implementations of DCMS (development of "cook-book" approach); training and documentation control of software. Requires: Technical Diploma in Data Processing and 3 years experience as Programmer/Analyst. Solid background with IBM S/38; expertise in all facets of wholesale distribution application software; analytical/programming abilities; ability to interview senior management personnel with respect to computer related requirements; background in project management and proven organizational and written and oral skills. \$3,000 per month. 40 hours per week. Job Site/Interview. Corona. Qualified applicants send resume or application letter to Job #MLU8574, P.O. Box 9560 Sacramento, CA 95823-0570. No later than January 1, 1986.

THE PENNSYLVANIA STATE UNIVERSITY
Computer Engineering

Applications are invited for tenure-track and visiting faculty positions at all levels. Candidates from all areas of computer engineering (hardware and software) will be considered. The computer engineering program at the Pennsylvania State University within the Department of Electrical Engineering, which has over 50 faculty members, and approximately 1500 undergraduate majors, 170 graduate students. Candidates should have a PhD in Electrical/Computer Engineering or related areas. There are 15 faculty members within the computer engineering program. Excellent teaching and research computing facilities are available within the department, college and at the university computing center.

Please send your letter of application, resume, or inquiries, together with three references to: T. Feng, Computer Engineering Program, Department of Electrical Engineering, 129 Electrical Engineering East, Box G, The Pennsylvania State University, University Park, PA 16802. Deadline for applications is March 31, 1987 or until suitable qualified candidates are selected. An Equal Opportunity/Affirmative Action Employer.

THE UNIVERSITY OF ALBERTA
Department of Computing Science

The Department of Computing Science is undergoing an extensive expansion in research initiatives. Applications are invited for two tenure-track positions at the Assistant/Associate Professor level. Responsibilities include research as well as teaching at the graduate and undergraduate levels. Candidates from all areas will be considered. Current hardware support includes an Amdahl 5870, a network of VAX 11/780's, well equipped microcomputer and workstation laboratories for graphics, VLSI, and AI research. Access to a Cyber 205 is available. Salary range is \$31,612 to \$50,630 and is commensurate with qualifications and experience. Send curriculum vitae, names of three references, and up to three reprints or papers. New Ph.D.'s should also include a copy of their transcript. Apply to: Dr. Lee White, Chairman, Department of Computing Science, University of Alberta, Edmonton, Alberta, T6G 2H1. Applications will be accepted until December 31, 1986. The University of Alberta is an equal opportunity employer.

UNIVERSITY OF CALIFORNIA, DAVIS
Faculty Positions in Electrical and Computer
Engineering/Computer Science

The Department of Electrical and Computer Engineering at UC Davis invites applications for tenure track positions at all ranks in all specialties of electrical engineering, computer engineering, and computer science. Of particular interest are faculty positions in the areas of integrated optics and optoelectronics, computer architectures, artificial intelligence, computer communications, and image processing. The department, with 40 faculty members and 150 full-time graduate students, is experiencing rapid growth. Our College is the nation's sixteenth largest producer of engineering Ph.D.'s in a University which has the twentieth largest extramural research funding. Salary and benefits are extremely attractive.

Davis is a pleasant, family oriented community, within easy driving distance to Silicon Valley, the Lawrence Livermore National Laboratory, San Francisco, the Pacific Ocean, and the Sierra Nevada Mountains.

We are seeking individuals with strong records of teaching and research with ambitious plans. Senior appointments require outstanding records of achievement, junior appointments must show evidence of great promise. All faculty are expected to have a strong commitment to teaching at all degree levels, and to demonstrate the ability to attract significant research support. The positions require a Ph.D. degree or equivalent, and are open until filled. Send a resume and the names of at least three references to:

Professor S. Louis Hakimi, Chair
 Department of Electrical and
 Computer Engineering
 University of California
 Davis, CA 95616

The University of California, Davis, is an equal opportunity/affirmative action employer.

GRAPHIC SOFTWARE ENGINEER

Design, develop and integrate interactive computer graphic software into Professional CAD/AM system. Resolve technical problem formulations; prepare flow charts, block diagrams. Have knowledge of advanced computation geometry with ability to analyze, review and rewrite programs to increase operating efficiency and/or adapt to new requirements. Ability to communicate effectively and use independent judgement in problem solving situation. Two years experience in CAD/CAM or related field. Will accept M.S. in lieu of experience. Must know UNIX, FORTRAN, C. \$600/wk. Job and interview in Burbank. Send ad and resume to Job #TC8091, P.O. Box 9560, Sacramento CA 95823-0560 not later than December 31, 1986.

AGRICULTURE ENGINEER

Will engage in feasibility studies of micro-computer based applications for solution of technical and administrative agribusiness problems such as optimization of resources, scheduling and program planning, providing crop management data, projections of crop yields, and maintaining records of agricultural activities. Work with computer hardware and software engineers to develop and present sales proposals to worldwide agribusiness customers. Req. BS in agriculture plus 3 years R&D experience in agribusiness. \$38,400 per year, 40 hours per week. Place of employment and interviews: Santa Clara, CA. Send this ad and a resume to job #MLU47829, P.O. Box 9560, Sacramento, CA 95823-0560, not later than December 31, 1986.

YORK UNIVERSITY
Faculty of Arts

Department of Computer Science. Tenure track and sabbatical replacement positions at the assistant, associate and full professor levels. A PhD in computer science or equivalent is required. All areas in computer science will be considered. Preferred areas are: networking and communications, operating system and computer architecture, graphics, parallel computation, artificial intelligence, database management and office automation. Duties include teaching and research. A proposal for research is currently under development. The Department's equipment includes: a VAX 8600 (VMS), and a VAX 11/750 (Unix), and Sun Workstations, as well as microcomputer and graphics laboratories. Faculty and students also have access to IBM 4381 (CMS, UTS) facilities. York University is located in Metropolitan Toronto and within easy reach of downtown Toronto which offers excellent cultural, recreational, and entertainment facilities. Send curriculum vitae and names of three references to Prof. E. Arjomandi, Chairman, Department of Computer Science, Faculty of Arts, York University, North York (Toronto), Ontario, Canada M3J 1P3. York University is implementing a policy of employment equity. Qualified women and men are invited to apply. In accordance with Canadian immigration requirements, priority will be given to Canadian citizens and permanent residents of Canada.

SENIOR SYSTEMS PROGRAMMER

Responsible for analyzing and installing software and updates; monitoring performance of software system and tune system when required; offering support to applications programming, computer operations, and users; maintaining a high level of technical competence in software programming, installation, maintenance, tuning of system and application software on minicomputers in a large network connected to mainframes. Must have knowledge in development of full screen applications using PL/I and Pascal computer languages under VMS/SP in an engineering environment. Must have 2 years college in the field of Computer Science or Electrical Engineering and 3 years experience in job or as Systems Designer. 40 hours per week, 8:30 am to 5 pm, \$41,300 per year. Job Site: Anaheim, CA. Send this ad and a resume to Job #TC8103, P.O. Box 9560, Sacramento, CA 95823-0560 not later than 12/31/86.

UNIVERSITY OF MASSACHUSETTS
Imaging Scientist

Faculty position for imaging scientist to do research and teach in new Biomedical Imaging Division. Individual desired whose area of interest would complement those of existing members of an active and growing biomedical imaging group composed of basic scientists and clinicians. Possible areas of interest of applicant: artificial intelligence; hardware design and development for image acquisition, analysis or display; development of mathematical methods for image processing and analysis. Must have image processing facilities available to the candidate. Joint appointment at nearby engineering school also possible. Rank and salary are negotiable depending on background and experience of applicant. To apply send by January 1, 1987, curriculum vitae and names of three references to: Dr. Fredric S. Fay, Director, Biomedical Imaging Group, University of Massachusetts, Medical Center, 55 Lake Avenue North, Worcester, MA 01605. University of Massachusetts Medical Center is an equal opportunity/affirmative action employer.

UNIVERSITY OF SOUTHWESTERN LOUISIANA
The Center for Advanced Computer Studies
Research Faculty Graduate
Fellowships/Assistantships In
Computer Science/Engineering

The Center for Advanced Computer Studies is a research center with programs leading to MS/PhD degrees in Computer Science and Computer Engineering. External grants/contracts support research in a variety of areas. The Computing Research Laboratory includes a 40-node Sun-3 network, a 6-processor Encore Multimax, 2 VAX 11/780s, logic development systems, laser printers, plotters, and numerous other equipment. Instruction utilizes a 3-processor Pyramid 90X network running UNIX and an IBM 3090 with a vector processor. Another well-equipped facility supports research in CAD/CAM. About 300 students are enrolled in computing graduate programs.

RESEARCH FACULTY: Applications are invited from persons holding PhD degrees with demonstrated research capabilities in Computer Science/Engineering. Persons appointed as Assistant Professors must hold PhDs in Computer Science/Engineering and have research potential. Associate Professors and Professors must hold PhDs and have an established research publication and grant record. The typical teaching load is 2 graduate-level courses per year and a continuing research seminar. Salaries range from about \$50,000 to over \$100,000 per year. Excellent support is provided for travel, equipment, research assistants, and professional activities. Thus, the necessary environment exists to achieve your professional goals. To apply, send a copy of your curriculum vitae and the names and addresses of at least 3 professional references. Applications will be considered until all positions are filled.

PHD FELLOWSHIPS: A number of PhD Fellowships valued at more than \$14,000 per year are available. They provide support for up to 4 years of study towards the PhD in Computer Science or Computer Engineering. Recipients also receive preference for low-cost campus housing. Completed applications must be received by 15 February 1987.

MS/PHD ASSISTANTSHIPS: More than 100 teaching and research assistantships are available to support students pursuing an MS or PhD degree in Computer Science or Computer Engineering. Stipends are valued at from about \$6,000 up to \$12,000 per academic year. Completed applications must be received by 1 March 1987.

APPLICATIONS: Dr. Terry M. Walker, Chairman, Faculty Search Committee, The Center for Advanced Computer Studies, Lafayette, LA 70504-4330.

An Affirmative Action/Equal Opportunity Employer.

WASHINGTON UNIVERSITY IN ST. LOUIS

The Department of Electrical Engineering at Washington University in St. Louis is seeking a new Chairman to guide an expansion of about one-third from its present eighteen person level; additional positions are anticipated from retirement over the next decade. A major gift to underwrite the opening phases of this development is already in hand. Persons desiring additional information should write or telephone Professor Fred J. Rosenbaum, Department of Electrical Engineering, Washington University, St. Louis, Missouri 63130 [314]889-6154. Washington University is always interested in superior applicants regardless of race, creed, color, gender or age.

OHIO STATE UNIVERSITY
Department of Computer and
Information Science

Applications are invited for faculty positions at all levels. A Ph.D. in computer science or a closely related field is required. Of special interest are candidates in the areas of artificial intelligence, computer graphics, data bases, parallel and distributed computing, software engineering, and VLSI design. Special research support packages are negotiable for highly qualified candidates. Research computing facilities within the Department include DEC 20/60, VAX 11/780, IBM 4341, AT&T 3B2, Xerox Lisp and XDE machines, Sun and HP workstations, and several experimental multi-computers (Intel Hypercube, BBN Butterfly, etc.). In addition to the University's computing facilities (IBM 3081-D, etc.) the department also has access to national networks including ARPANET and CSNET.

To apply, please send application and resume, including a statement of research and teaching interests and he names and address of at least three references, to Prof. Ming T. (Mike) Liu, chairman of Faculty Search Committee, Department of Computer and Information Science, The Ohio State University, 2336 Neil Avenue, Columbus, Ohio 43210-1277 (CSNET:LIU@OHIO-STATE). The Ohio State University is an equal opportunity/affirmative action employer.

THE UNIVERSITY OF TEXAS AT AUSTIN

Electrical and Computer Engineering Department is accepting applications for tenure and tenure-track faculty positions in computer engineering or microelectronics. Qualifications must be commensurate with rank and include an outstanding academic record, significant achievement in original research, sincere interest in teaching, and a doctorate in electrical or computer engineering/science. Duties will include teaching undergraduate and graduate courses. Interested applicants are invited to send resumes which detail their past professional accomplishments, and which include the names and addresses of three references, to Dr. Edward J. Powers, Chairman, Department of Electrical and Computer Engineering, The University of Texas at Austin, Austin, Texas 78712-1084, an Equal Opportunity/Affirmative Action Employer.

UNION COLLEGE

The Department of Electrical Engineering and Computer Science invites applications for the position of Assistant/Associate Professor in Computer Science. There are openings for two tenure-track positions. An applicant should have a doctorate with major emphasis in C.S. Good teaching is expected and supported. The College maintains regular budgets for faculty research and travel.

Areas of interest include, but are not limited to, artificial intelligence, data base systems, algorithm design, and software engineering. Union College academic computing resources include a VAX cluster (with three 785's, a 780, and a 750) running VMS 4.4 and UNICOS, as well as the department's computer science lab with a VAX-11/750, ten DEC Pro350's, and two AT&T 3B2 machines all running the UNIX operating system.

Applicants should write: Dr. David G. Hannay, Co-chairman, EECIS Department, Steinmetz Hall 210, Union College, Schenectady, NY 12308.

An Equal Opportunity/Affirmative Action Employer.

UNIVERSITY OF MINNESOTA
Computer Science Department

The Computer Science Department invites applications for regular, visiting and adjunct faculty positions at all ranks and in all areas of computer science. The department also has distinguished professorships available. All regular faculty positions require an earned doctorate, and involve teaching, research and service. Assistant Professors must have demonstrated research and teaching ability. For tenured Associate and Full Professorship positions, candidates must have demonstrated effectiveness in teaching and at least a reputation in scholarly research. For Associate Professorships, a minimum of three years teaching and research is required; for Full professorship a minimum of six years teaching and research is required or an appropriate combination of these.

Non-regular positions as teaching specialists and lecturers are also available. Candidates for lecturer must have an earned doctorate, and candidates for teaching specialist must have a Master's degree in computer science or related fields.

While applicants from all areas of computer science are sought, we are especially interested in applicants with strong research interests in the following areas: artificial intelligence, robotics, computer architecture, computer graphics, operating systems, software engineering, software systems, large scale scientific computing and supercomputing.

Minnesota is a major center of the computer industry. The Computer Science Department currently has 24 full-time faculty and numerous adjunct faculty. The activities of the department are augmented by the activities of the Minnesota Supercomputing Institute, the Microelectronics and Information Sciences Center, the Institute for Mathematics and its Applications, and the Center for the History of the History of Computer Science.

University-wide computing facilities include a CRAY-2, a CYBER 205, an ENCORE multimax, and several other CYBERs and VAXs. The department of Computer Science operates a VAX-11/780, an MV-10000, clusters of SUN and HP workstations, a network of Apollo workstations, a 16 node and a 64 node NCUBE hypercube computer. The department is on CSNET and ARPANET.

Applications should be received by February 28, 1987. This deadline will be extended in case all available positions are not filled. The department will begin to review applications after January 1, 1987. All degree requirements for the Ph.D. must be completed before the starting date of the appointment.

Applicants should send their resume and the names of at least four references to:

Professor Sarraj Sahni, Chair, Faculty Search Committee, Computer Science Department, 136 Lind Hall, University of Minnesota, 207 Church Street SE, Minneapolis, MN 55455.

The University of Minnesota is an equal opportunity educator and employer and specifically invites and encourages applications from women and minorities.

UNIVERSITY OF CALIFORNIA, SANTA CRUZ
Computer Engineering
Computer and Information Sciences

Information Sciences at the University of California, Santa Cruz, invite applications for positions as Assistant, Associate or Full Professor, as appropriate, effective July 1, 1987. The Division of Natural Sciences at UCSC is expanding its program in Computer and Information Sciences and its new program in Computer Engineering.

LOYOLA UNIVERSITY

Loyola University, New Orleans invites applications for assistant professor tenure track positions in the Department of Mathematical Sciences. Qualifications needed:

(1) A doctorate in computer science, or computer engineering. ABD candidates in computer science or computer engineering will be considered with the understanding that the degree must be conferred within two years of appointment to the faculty. Duties including teaching upper-level undergraduate computer science courses such as systems programming, survey of programming languages, computer graphics, database management systems design, automata theory, and control systems.

(2) A doctorate in mathematics with a master's degree in computer science, or computer information systems. Duties include teaching lower-level undergraduate computer science courses such as introduction to computer science, programming techniques, and data and file structures, and lower-level undergraduate mathematics courses.

Individuals must have a commitment to quality teaching and research competence. Responsibilities include teaching three or four undergraduate computer science courses per semester (depending on research activities), research, advising majors, and contributing to the development of the department, the college, the university, and the community. Contracts are on a nine month basis with summer teaching possible.

The Department of Mathematical Sciences has 16 full time faculty members and more than 200 majors. The degrees offered by the department are: The B.S. degree in Mathematics, the B.S. degree in Computer Information Processing, and the B.S. degree in Computer Science. Academic computing facilities include an HP 3000/48 and a VAX 11/750. Departmental computing facilities include two microcomputer labs and a digital electronics lab. A variety of microcomputer equipment is available.

Loyola is a small liberal arts university known for quality education and personal interaction between students and faculty. New Orleans is a large sunbelt port city famous for excellent cuisine, jazz, and Mardi Gras. Loyola is an Equal Opportunity/Affirmative Action Employer. Women and minorities are encouraged to apply.

To be assured of full consideration, complete applications must be received by March 5, 1987.

Late applications may be considered if the position is still open after those applying earlier have been considered. A complete application consists of a current resume, a transcript of all graduate level course work, and three letters of recommendation sent to:

Mathematical Sciences Search Committee, Loyola University, Box 104, New Orleans, LA 70118

WESTERN WASHINGTON UNIVERSITY

Funding permitting, Computer Science will have two tenure-track openings for the academic year starting Sept., 1987: Ph.D. In Computer Science or closely related field is required. There is need to staff the mainline undergraduate courses and to strengthen the masters' program.

For more information, please contact James Johnson, Chairman, Computer Science Dept., Western Washington University, Bellingham, Washington 98225; CSNET address: Johnson %wwu@csnet-illu.arpa WUWU is an EO/AA employer.

ARIZONA STATE UNIVERSITY

Applications are invited at all levels for tenure track faculty positions in computer science and engineering. While exceptional candidates from all areas will be considered, special consideration will be given to candidates in the areas of VLSI architectures, distributed and database systems, languages, and compilers, computer aided geometric design, computer graphics, and artificial intelligence. Consistent with the high technology base of this rapidly developing community, the State and the University have entered the second phase of a nationally recognized program in Engineering Excellence, Computer Science and Engineering is its primary target area of this program. A new research center was occupied in 1984, and plans are well underway for the construction of a new design center to complement it. A wide variety of university, college, and departmental computers are available including VAX8600's, 780's, 750's running VMS and Unix, IBM3085, 4381, 4341, and PCs, Harris HCS-7, multiprotocol networking, and several student microcomputer laboratories. The ASU Research Park was completed in July and is filling rapidly, thus providing additional opportunities for the faculty.

Please send a vitae and names of at least three references to Dr. Ben Huey, Faculty Search Committee, Department of Computer Science, Arizona State University, Tempe, AZ 85287. First deadline for applications is February 31, 1987 and thereafter until positions are filled. Affirmative Action/Equal Opportunity/Title IX Employer

GEORGIA INSTITUTE OF TECHNOLOGY

The School of Information and Computer Science invites applications for faculty positions at all levels. Applicants should have a commitment to teaching and should show a record of outstanding research accomplishments or expect to receive a Ph.D. degree by Fall 1987 and show high potential for research. The School seeks applicants to strengthen its capabilities in all areas of current research activity, especially artificial intelligence, software engineering, distributed computer systems, and also in computer graphics, programming languages, theoretical computer science, human factors, VLSI systems, data communications and computer networks, database systems, and computer architecture. Very competitive salaries are offered.

The School is well supported by the Institute. It has 24 faculty members and is currently experiencing substantial faculty growth. Its educational activities include an undergraduate program accredited by the Computing Sciences Accreditation Board, Inc., a Masters program with 150 students and a Ph.D. program with over 70 students. Well equipped separate laboratories support research and education. High-speed local area networks interconnect all major campus laboratories and provide access to CSNet and other national networks.

Georgia Tech is located in Atlanta, which experiences a mild sunbelt climate. It is the center of commerce in the Southeast, offering a diverse economy and good employment opportunities in all professional areas. Atlanta offers good cultural and recreational opportunities, extremely attractive residential neighborhoods, and affordable housing.

Candidates should send complete resumes and names of at least three references to: Professor Nancy D. Griffith, Chairman, Faculty Search Committee, School of Information and Computer Science, Georgia Institute of Technology, Atlanta, Georgia 30332. Georgia Tech is an equal opportunity employer.

CALIFORNIA STATE UNIVERSITY, HAYWARD Department of Mathematics and Computer Science

The department is now seeking applicants for tenure track Assistant Professor positions in Computer Science beginning Fall, 1987. Particularly well qualified, experienced individuals with the capability to exercise leadership in the program may apply for appointment at the rank of Associate Professor.

Applicants should have the Ph.D. in Computer Science, or in a related field with computer science experience. They should have a consistently well qualified, experienced individuals with the capability to exercise leadership in the program may apply for appointment at the rank of Associate Professor. Applicants should also exhibit the competence and potential to engage in significant professional activities, including research and publication. All areas of specialization will be considered. The interests of the present faculty include a wide range of theory, software design, and hardware design topics.

CSU, Hayward is located in the hills above, and overlooking, the eastern shore of the San Francisco Bay. More than 12,000 students attend this university, which has outstanding programs in arts, letters, science, and business. The Department of Mathematics and Computer Science enrolls nearly 1300 majors in its three degree programs: B.S. in Computer Science, B.S. in Mathematics, and M.S. in Mathematics. An M.S. in Computer Science is being planned.

The department has a Pyramid 90x minicomputer and four Sun3-160 color workstation computers on Ethernet, a graphics lab, an interactive classroom, a digital and a microprocessor lab, and numerous IBM PC's for faculty use. In addition, the campus has a Cyber 170, a PDP 11/44, a PRIME 9720, and numerous PC's for student use.

Interested applicants should send a resume and the names of three references to

William R. Nico, Chair
Department of Mathematics and Computer Science
California State University, Hayward
Hayward, CA 94542
UUCP: ...ucbvax11-crc@caunixico

Applications received by February 20, 1987, will be assured full consideration. Applications will be accepted as long as positions remain available.

California State University, Hayward, is an Equal Opportunity/Affirmative Action employer and encourages applications from women and men of all ethnic backgrounds and physical abilities.

UNIVERSITY OF WISCONSIN—MADISON Faculty Positions

The Department of Electrical and Computer Engineering invites applications for tenure and tenure-track faculty positions. A Ph.D. degree is required, and successful candidates are expected to participate in both teaching and research activities. Applicants in all areas of computer engineering are invited to apply, but the following areas are of special interest: computer architecture, computer networks, large-scale computational methods, real-time computer applications, design automation, VLSI circuit design, and engineering applications of artificial intelligence. Rank and salary will be commensurate with qualifications and experience. Send resume and names of three references to: J. Leon Shohet, Chairman, Department of Electrical and Computer Engineering, University of Wisconsin-Madison, 1415 Johnson Drive, Madison, WI 53706, an equal opportunity/affirmative action employer.

UNIVERSITY OF CALIFORNIA SANTA BARBARA

THE UNIVERSITY OF CALIFORNIA AT SANTA BARBARA invites applications for four tenure track positions at senior professional levels in the Department of Computer Science, and for three additional positions in closely related areas, including computer engineering. The Department of Computer Science is in a rapidly expanding College of Engineering that has achieved national prominence in many areas of research. UCSB and the College of Engineering are highly committed to augmenting its Computer Science Department in the department that has both excellence and national visibility. The senior positions in Computer Science currently available are in three major areas of research in which the Department of Computer Science intends to achieve its greatest strength. These areas are: machine vision and motion planning; parallel computing; and software engineering. These positions will provide opportunities for successful candidates to guide the growth and development of the Department in these areas. Strong material support will be given to such positions. "State of the Art" laboratories are currently being planned for instructional purposes in these areas. Research facilities in these areas will be made available and tailored to the needs of successful applicants. Graduate student support will be given high priority.

Interactions with various research groups on campus (Center for Robotic Systems, Cognitive Science Program, Center for Scientific Computing, Institute for Theoretical Physics, etc.) will be strongly encouraged and supported. Applicants must possess a doctoral degree and an extremely strong record of research accomplishments. Teaching experience is desirable. Outstanding younger candidates are also encouraged to apply. Deadline for receipt of applications is February 15, 1987. Send resume and names of at least eight references to:

Chairman
Computer Science Faculty Search
Committee
Department of Computer Science
University of California
Santa Barbara, CA 93106

The University of California is an Equal Opportunity/Affirmative Action Employer.

UNIVERSITY OF HOUSTON

Applications are invited for tenure track faculty positions in the Department of Computer Science starting September 1987. Areas of special interest include but are not limited to: artificial intelligence, computer architecture, operating systems, programming languages and software engineering. Rank and salary are open and competitive. The Department is interested in expanding its research program and particularly welcome applicants for senior positions. Applicants should have Ph.D. in Computer Science or a closely related area, and a strong commitment to research and teaching. Candidates for senior positions should also have a distinguished research record. The Department offers Ph.D., M.S., and B.S. in Computer Science. Departmental research facilities include a network of VAX 11/780 and VAX 11/750's, a network of AT&T 3B20 and 3B2's along with access to other computer facilities in the University Computer Center as well as supercomputers via remote access terminals. Send resume and names of professional references to: Dr. Willis K. King, Chairman, Department of Computer Science, University of Houston, Houston, Texas 77004. An Equal Opportunity/Affirmative Action Employer.

UNIVERSITY OF FLORIDA
Computer & Information Sciences
Department

Applications and nominations are invited for a person to chair a young, dynamic, and growing department at a land grant university in the beautiful state of Florida. Our rapidly expanding department now numbers 18 doctorate faculty and offers programs leading to the B.S., M.S., and Ph.D. degrees. It has well funded research programs in database systems, information science, digital systems, software engineering, and image processing.

Department facilities include a Gould Power-note 9080 (4.2 BSD UNIX), four Vax 11/750's (4.2 BSD UNIX and VMS) supporting CSNET, two IBM Series/1, three Apollo workstations, five Sun workstations and numerous (50+) smaller computers. A local area network provides access to all the computer systems throughout the College of Engineering. The department and all other computer-related research activities and programs have recently moved into a new 50,000 sq. ft. engineering building with space for future expansion.

Qualifications include an earned doctorate in computer science or a related field, ability to provide strong leadership for research and academic programs and credentials appropriate for an appointment as a full professor. Appropriate administrative experience is desirable. Twelve-month appointment begins on or about August 1, 1987. Salary will be commensurate with rank and qualifications.

This search will be conducted in compliance with Florida's "government in the Sunshine" law. Qualified applicants should send their resumes and names of three references to Dr. Stanley Y.W. Su, Chairman, Search Committee, Computer and Information Sciences Department, 301 Computer Sciences and Engineering, Gainesville, FL 32611, CSNET address: su@ui. Closing date: February 15, 1987 or until position is filled.

An Equal Opportunity/Affirmative Action Employer.

UNIVERSITY OF HONG KONG
SENIOR LECTURER/LECTURERS IN
COMPUTER STUDIES
(Re-advertisement)

Applications are invited for posts of Senior Lecturer/Lecturer in Computer Studies in the Centre of Computer Studies and Applications. Candidates should have a higher degree in Computer Science, Computer Engineering or Information Systems, and a strong interest in both teaching and research. Preference will be given to applicants with research experience in graphics, information systems or computer architecture. Consideration may be given to applications for appointment on a short-term basis (please specify period). A certain amount of outside practice is permitted.

Annual salaries (superannuation) are on the scales: Senior Lecturer HK\$274,960-369,360 (9 points), Lecturer HK\$176,880-295,680 (11 points) (US\$1 = HK\$7.80 on 10-9-86). Starting salary will depend on qualifications and experience. At current rates, salaries tax will not exceed 17% of gross income. Housing benefits at a rental of 71% of salary, children's education allowances, leave, and medical benefits are provided for permanent appointees.

Further particulars and application forms may be obtained from the Secretary General, Association of Commonwealth Universities (Apsu), 36 Gordon Square, London WC1H 0PF, or from the Appointments Unit, Registry, University of Hong Kong, Hong Kong, Class 31 December 1986.

THE UNITED STATES
NAVAL ACADEMY

Applications are invited for tenure-track positions in the Computer Science Department. An earned Ph.D. in Computer Science or related field is required. The applicant must have a strong interest in combining teaching with research in a very high quality undergraduate program. Previous teaching experience is preferred. Applicants should be able to teach upper division mainstream computer science courses. Civilian faculty receive Excepted Civil Service appointment with initial appointment normally for three years. Salary commensurate with overall experience. Excellent opportunities for research in the local area.

The Computer Science Department offers a major in Computer Science leading to a Bachelor of Science degree. The curriculum follows the latest recommendations of the ACM Curriculum Committee.

Interested persons should send their vitae to: Chairperson, Computer Science Department, United States Naval Academy, Annapolis, Maryland 21402 (phone (301) 267-3060). Deadline for receipt of applications is 1 February 1987; however, applications will be considered until the positions are filled.

The United States Naval Academy is an Affirmative Action Equal Opportunity Employer.

THE UNIVERSITY OF TEXAS AT ARLINGTON

The Department of Computer Science Engineering (CSE) at the University of Texas at Arlington (UTA) invites your application for tenure-track or visiting faculty positions in all areas of computer science and computer engineering. Rank is open. An earned doctorate or equivalent and commitment to teaching and scholarly research is required. Openings are expected for June, or September 1987. Our department offers baccalaureate, masters, and doctoral programs. Excellent computing facilities are available including personal computers in each CSE faculty office. UTA is in the Dallas/Fort Worth metropolitan area near a large concentration of high technology industries. Interested persons should send a resume to Professor Bill D. Carroll, Professor and Chairman, Computer Science Engineering Department, P.O. Box 19015, The University of Texas at Arlington, Arlington, TX 76019. Phone (817) 273-3785.

The University of Texas at Arlington is an Equal Opportunity Affirmative Action Employer.

UNIVERSITY OF MARYLAND

The Electrical Engineering Department of the University of Maryland College Park has openings for regular faculty positions in computer engineering. Candidates for the rank of Assistant Professor should have a high potential for both teaching and research. Candidates for the rank of Associate Professor and Full Professor should have distinguished records in research and a strong interest in educational programs. The department conducts research and educational programs in all areas of computer engineering. Although applications in all areas of computer engineering will be considered, networking, operating systems, computer security, architectures for parallel computing, and VLSI design are currently the highest priority areas for adding new faculty. Application, including resume, list of publications, list of references, identification of technical area for which the candidate wishes to be considered, should be sent to Dr. William W. Dettler, Chairman, Electrical Engineering Department, University of Maryland, College Park, Maryland 20742. The University of Maryland is an equal opportunity, affirmative action employer.

WRIGHT STATE UNIVERSITY
Computer Science & Engineering

Applicants are invited for full-time positions in Computer Science and/or Computer Engineering at all faculty levels. There is particular interest in professors who have sufficient experience, currency, and depth to direct doctoral students and research dissertations in a PhD program in Computer Science & Engineering. The Department is also interested in PhD's with less experience, but who have a commitment to research and teaching in a PhD granting program. Rank and competitive salaries are determined by qualifications and experience.

The State supported university is located in a high technology environment among industrial/military research and development facilities. An associated State assisted research park to foster basic and applied industrial/military/University research is active and growing impressively. The University has identified Computer Science & Computer Engineering an area of highest priority for continued development.

Department strengths include a large faculty, extensive laboratory facilities, research programs, industrial/military support, degree programs in both computer science and computer engineering, and large student populations at graduate as well as undergraduate levels.

Successful candidates for tenure-track positions should have a Ph.D. in Computer Science, Computer Engineering, or equivalent background, and have demonstrated a strong research, as well as teaching, commitment. Any non-tenure-track positions will be filled by candidates with strong Computer Science or Computer Engineering credentials. At least a Master of Science Degree in the field is desired.

Please submit detailed resumes including names of three references to: Professor Larry A. Crum, Chair, Department of Computer Science, Wright State University, Dayton, Ohio 45435. Reviewing for tenure-track positions will begin December 1, 1986 and for non-tenure-track positions on March 1, 1987. Reviewing will continue monthly until positions are filled or until September 1, 1987. Salary will be commensurate with rank and qualifications.

An equal opportunity/affirmative action employer.

STATE UNIVERSITY OF NEW YORK
AT BUFFALO

The Department of Computer Science is seeking faculty at the level of Assistant Professor to begin Fall, 1987. Applicants must have a Ph.D. in Computer Science, or a related field, and superior research ability.

The University at Buffalo is the largest public university in New York and New England, with approximately 26,000 students enrolled in 93 undergraduate, 101 master's, and 94 doctoral programs, including law, medical, and business schools.

The Department offers BA, BS, MS, and PhD degrees, with controlled undergraduate enrollment. Departmental computing facilities include a VAX 11/780, two VAX-11/750s, six SUNs, six Ixms, and several image processing/graphics systems. Present research areas include artificial intelligence, computer vision, numerical analysis, parallel algorithms, theory of computation, VLSI, etc. The Department is expanding, with additional faculty lines committed annually. Salaries are extremely competitive.

Resumes and names of four references should be directed to: Prof. Sargur N. Srihari, Chairman of Search Committee, Department of Computer Science, 226 Bell Hall, SUNY at Buffalo, Buffalo, NY 14260 (CSnet: srihari@buffalo). SUNY is an affirmative action/equal opportunity employer.

**CALIFORNIA STATE UNIVERSITY,
FULLERTON**
Announcement of Open Faculty Positions

The Department of Electrical Engineering at California State University, Fullerton, seeks applications for tenure-track positions and lecturers, non-tenure-track positions, starting in February and August 1987. Rank, salary commensurate with qualifications. Applicants must hold a Ph.D. in Electrical Engineering and/or Computer Engineering with specialization in computers, communications and signal processing, systems, robotics or VLSI design.

California State University, Fullerton is a major urban university in Orange County, located close to many major high-technology corporations whose levels of sophistication in engineering are world renowned. As a result of this environment, the department currently serves over 1300 majors at the B.S. and M.S. levels. The department has curricular strength in the areas of communications, signal processing, digital systems, control systems, systems engineering, solid state electronics and electromagnetism. CSUF stresses the importance and value of teaching and research excellence. Successful applicants must be motivated to that end, and have the desire and ability to clearly communicate in the classroom/laboratory environment. Research is strongly emphasized and is an important element of a faculty member's overall scholarly and creative activity.

A new engineering building addition, essentially doubling the classroom office and laboratory capacity of engineering, is expected to be occupied in the Fall of 1988. Research is in state-of-the-art laboratories in robotics, VLSI design, digital control systems, microwave and computer aided design.

Candidates must indicate interest in tenure-track or lecturer positions or both. Minorities and women are particularly encouraged to consider joining this dynamic department. Research is emphasized and three references to Dr. John Clymer, Chair, Faculty Search Committee, Electrical Engineering Department, California State University, Fullerton, CA 92634. Applications accepted until positions are filled. CSUF is an Affirmative Action/Equal Opportunity employer.

SYSTEMS ENGINEER V

Responsible for analyzing data-processing requirements of clients in the U.S. and in Latin America in order to determine, install, test, and support electronic data-processing system that will provide required system capabilities. Must have in-depth knowledge of testing tools such as Core-DUMP readings and manufacturers utility programs; and, understanding of environmental factors involved in system layout. Position also requires demonstrated experience in the following computer skills: IBM 308X, 303X, and 4341 and Xerox Electronic Printing Systems 9700 and 8700 hardware; MVS/XA, JES2, TCM, TSO/ISPF-PDF, OS/VS1, VM/SP, VM/CMS, OS/IMS JCL, CICS/VS, UCC/LTMS, UCC2/DUO, UCC11/RESTART, OS/IMVT HASP, OS/IMFT HASP, DOS/VS, POWER/VS, XICS, EPIC, PLD, HFDL, XJCF, Dynam/TMS/DASD and XPGS operating system software; and, Assembler, COBOL, PLI, FORTRAN, and RPGII programming languages. Must also have Bachelor's or equivalent in computer science or data processing and four years experience in position or in occupation relating to computer systems engineering or analysis, or if degree is unrelated, must have 6 1/2 years with same experience. Salary \$35,000 per year, 40 hour work week. Place of employment and interview: Los Angeles, CA. Send ad with resume to Box #FC 8074, P.O. Box 9560, Sacramento, CA 95823-0560 no later than December 31, 1986.

UNIVERSITY OF NEBRASKA-LINCOLN
Computer Science and Computer Engineering
Faculty Positions

Applications are invited for several tenure-track positions in Computer Engineering starting August, 1987. Rank and salary open with preference given to Assistant Professors. Requires Ph.D. in Computer Science, Computer Engineering or a related field. Must have strong commitment to research and teaching.

Seeking candidates from all areas of computer science and computer engineering, particularly welcome expertise in any of the following areas: computer architecture, parallel processing, VLSI, operating systems, computation, analysis of algorithms, artificial intelligence and programming languages.

The department has in place rigorous programs in computer science research and education and is planning growth in the direction of computer engineering. Computer Science is intimately tied to two colleges: Arts & Sciences and Engineering & Technology, and offers degrees in both colleges. As the primary campus for research and graduate studies in Nebraska, UNL provides a modern computing environment to all its faculty. In particular, computer science faculty have access from their offices to departmental and campus computers and to CSNET. Currently about 62 Masters and 10 Ph.D. candidates are enrolled in the graduate program. There are 15 full-time faculty with research interests in analysis of algorithms, artificial intelligence, computer architecture, fault-tolerant computing, VLSI, data bases, programming languages, formal languages, combinatorics, and discrete mathematics, symbolic and algebraic computation, simulation, networks, theory of computation, performance evaluation, and human-computer interaction.

Submit letter of application and vitae by December 1 (or until positions are filled) to: Chair, Search Committee
Department of Computer Science
University of Nebraska-Lincoln
115 Ferguson Hall
Lincoln, Nebraska 68588-0115

Affirmative Action/Equal Opportunity Employer

LOUISIANA STATE UNIVERSITY
Faculty Position in Image Processing

The Department of Electrical and Computer Engineering and the Remote Sensing and Image Processing Laboratory invites applications for an assistant faculty position. The person would teach computer vision and computer engineering courses in the Department. Research in computer vision would be conducted with the Remote Sensing and Image Processing Laboratory (RSIP). The position is attractive regarding a teaching load balanced to provide the ability to develop an academic program and provide time and resources for developing a high quality research program in computer vision. The RSIP Laboratory has well qualified staff and modern computing and image processing equipment. A number of peripherals including Symbolics, Sun, and International Imaging workstations are available. A comprehensive image processing system has been implemented in Berkeley Unix 4.3. The appointment will be made at a rank appropriate with the qualifications of the applicant. Interested applicants should apply by submitting a resume, statement of interests, and names and addresses of three professional references. Applications that apply by January 31, 1987 will be given preference. Apply to: Professor Charles A. Harlow, Director, RSIP Laboratory, 150 Electrical Engineering Building, Louisiana State University, Baton Rouge, Louisiana 70803-5901. LSU is an Equal Opportunity Employer.

BUCKNELL UNIVERSITY

Bucknell University invites applications for tenure track faculty positions at the assistant or associate professor level. A Ph.D. in Computer Science or related discipline is normally required. Applicants should demonstrate both an interest in and a promise of developing excellence in teaching and research. All research areas considered. First year research and summer support funds available.

The computing environment includes VAX 11/780 VMS, VAX 11/750 UNIX, Sun3 and Apollo workstations, DG Eclipse network lab, HP instrumentation lab, VAX Station II, and a Honeywell Level 66 DPS mainframe. Please send a curriculum vitae and the names of three references to Gary Haggard, Dept. of Comp. Sci., Bucknell Univ., Lewisburg, PA 17837 or HAGGARD@BUCKNELL.BITNET. Applications will be considered beginning November 15. Appointments for January 1987 are possible. Applications from women and members of minority groups are encouraged.

UNIVERSITY OF CALIFORNIA, RIVERSIDE

Applications are invited for a tenure-track or tenure position in Computer Science beginning Fall 1987. Candidates must have demonstrated excellence in research and teaching. Research specialties in all areas of Computer Science will be considered but we are particularly interested in research areas in Computer Systems or Computer Methodology and Applications. The position is open as to the level of appointment. Applicants should send a curriculum vitae and see that at least three letters of recommendation are sent to:

Professor Theodore J. Barth, Chair
Computer Science Search Committee
Department of Mathematics and Computer Science
University of California
Riverside, CA 92521

University of California, Riverside, is an Affirmative Action/Equal Opportunity Employer.

UNIVERSITY OF CALIFORNIA
SANTA BARBARA
Electrical and Computer Engineering

Applications are invited for three tenure track faculty positions in computer engineering. One position is in the area of design automation (CAD tools, techniques, algorithms, or expert systems) to support and/or validate VLSI-based designs. The remaining positions may be filled from any specialty within computer engineering, with preference given to applicants experienced in the fields of parallel computing, very high-speed digital circuit design/analysis, or fault-tolerant computing. The positions start at the rank of Assistant Professor, although higher level appointments are possible for individuals with outstanding records. Normally, completion of a doctorate is required at the time of the appointment. Candidates should have an outstanding research potential or a distinguished research reputation, the ability to attract external research funding, and a strong commitment to teaching at the undergraduate and graduate levels. Applicants should send their resume, copies of recent publications, and the names of at least four professional references to: Chairman, Computer Engineering Faculty Search Committee, Electrical and Computer Engineering, University of California, Santa Barbara, CA 93106, (805) 961-3821. Applications will be received until the positions are filled. An Equal Opportunity/Affirmative Action employer.

UNIVERSITY OF CALIFORNIA, LOS ANGELES
Computer Science Department

The Department of Computer Science at the University of California, Los Angeles, invites applications for tenure-track positions at the Assistant Professor level in Computer Science beginning in July 1987. Applications are also welcome from highly distinguished candidates at the senior level. Applicants should possess the Ph.D. in Computer Science by July 1987.

Quality is our key criterion for selecting applicants. We expect them to have a strong commitment to both research and teaching and an outstanding record of research for their level. It is important that they have potential for continued excellence in university research.

We seek applicants in any mainstream area of Computer Science and is an particularly welcome those with research strength in software related areas.

Interested applicants should send a letter of application, a resume, and three references to: Professor Gerald Estrin, Chair, Computer Science Department, Boelter Hall 3713, University of California, Los Angeles, CA 90024.

The University of California is an Affirmative Action/Equal Opportunity Employer.

HARVEY MUDD COLLEGE
Engineering and Computer Science

Applications are invited for a tenure track position in the Engineering Department. Appointment at the Assistant Professor level is anticipated. Applicants should have an engineering background with graduate study in computer engineering or electrical engineering and computer science and experience in both hardware and software design and development. Responsibilities will include teaching in a unified engineering curriculum, developing courses and supervising industrially sponsored projects in the Engineering Clinic. Continuing professional growth and development through research or consulting is expected; excellent opportunities exist in the local area. A doctorate in engineering or computer science is required. Industrial experience is desirable. Reply to: Professor James E. Monson, Chairman, Search Committee, Harvey Mudd College, Claremont, CA 91711. Harvey Mudd College is an equal opportunity/affirmative action employer.

NORTH CAROLINA STATE UNIVERSITY

The Department of Computer Science invites applications for junior level tenure track positions. Applications are welcome from all areas, but are especially encouraged in the areas of operating systems, artificial intelligence, database systems, programming languages, architecture, and parallel processing. Candidates must have a Ph.D. in computer science or a related discipline and have a strong interest in both research and teaching.

Our department has research ties to organizations in the Research Triangle Park including MCNC, the Microelectronics Center of North Carolina. On campus we have research opportunities with several nationally recognized centers.

Applicants should submit a resume including research interests and three references to:

Dr. Edward W. Davis
Recruitment Committee
Department of Computer Science, Box 2606
North Carolina State University
Raleigh, NC 27695-8206

Immigration status of non-US citizens must be stated in the application. NCSU is an equal opportunity/affirmative action employer.

WORCESTER POLYTECHNIC INSTITUTE

The Computer Science Department invites applications for tenure track faculty positions at all levels from candidates in all areas of specialization. Candidates should have a Ph.D. in Computer Science and a strong interest in both research and teaching.

Worcester Polytechnic Institute emphasizes quality in the undergraduate learning experience and is committed to an innovative project-oriented teaching environment. The quality of the undergraduate computer science degree has been recognized by the recent granting of accreditation by the Computing Sciences Accreditation Board.

The current goal of the Institute is to enhance our graduate program and improve research activities. The department seeks qualified candidates who will help us achieve these objectives. WPI is located close to the center of Massachusetts' minicomputer industry, and excellent opportunities exist for cooperative research and consulting.

The Department has 12 full-time faculty with 200 undergraduates and 35 graduate students in our M.S. and Ph.D. programs. Department equipment includes three VAX 750's, an MV8000, twenty PDP-11's, and 40 PC's. Much of this equipment is networked via two ethernet cables and is also connected to other campus facilities which include a DEC 2060 and numerous VAX's. The Institute is committed to a new information Science building and full campus networking facilities in the near future.

Located only 45 miles from Boston, Worcester is a small city of 180,000 which has recently experienced a renaissance. It has eight colleges and universities and a rich variety of cultural activities.

Please send a resume to Prof. R.E. Kinicki, Head, Department of Computer Science, WPI, Worcester, MA 01609.

WPI is an Equal Opportunity/Affirmative Action Employer.

UNIVERSITY OF ILLINOIS AT
URBANA-CHAMPAIGN

The Department of Psychology, Mechanical/Industrial Engineering, Computer Science, and the Institute of Aviation at the University of Illinois at Urbana-Champaign are seeking to fill faculty positions for Fall, 1987, for their joint program in engineering psychology and human factors. The candidates should have research and teaching interests in any of the areas that can be viewed as related to the interface between people and computers. This encompasses areas such as the design of displays, the process of computer programming, the modeling of human computer interactions, human-machine modeling and simulation, computer-aided design and manufacturing, automation, applied cognitive psychology, and decision making. The faculty member is expected to be appointed in one or more of the above departments, with the opportunity to associate with the National Center for Super Computing Applications. A candidate is expected to support a strong research program conducted in the facilities of these departments. Undergraduate and graduate teaching will be required. Applicants are expected to have Ph.D. by August, 1987. Rank and salary of the position are subject to negotiation.

To ensure full consideration, applications with resume and three letters of reference should be sent by March 1, 1987 to Christopher Wickens, Chair, Human Engineering Search Committee, Department of Psychology, 603 E. Daniel, Champaign, IL 61820 (217) 333-3162. The University of Illinois is an Affirmative Action/Equal Opportunity Employer.

OSBERLIN COLLEGE
Computer Science Faculty Position

The Computer Science Program at Oberlin College invites applications for a full-time continuing faculty position in the College of Arts and Sciences. The position, which has been authorized for an initial term of 4 years beginning July 1, 1987, will carry a rank and salary commensurate with experience and qualifications.

The incumbent will teach courses in the general area of Computer Science, including at least one course in the appointee's area of specialization. In addition, he or she will have the opportunity to work with students in the Honors Program in Computer Science and will be expected to maintain a scholarly research program.

Among the qualifications required for appointment is the Ph.D. degree in hand or expected by September, 1987. Candidates must demonstrate interest and potential excellence in undergraduate teaching. Previous experience as a teacher is desirable.

The appointee will participate fully in the newly established major program in Computer Science which currently has an authorized staff size of four faculty positions. Oberlin offers an innovative Computer Science Major rooted in the liberal arts model curriculum, with some emphasis towards symbolic programming and artificial intelligence. A VAX 11/750 running UNIX with a full 8 megabytes of main memory and 431 megabytes of disk storage is dedicated to the computer science program. Additional software available on this system include two Scheme systems, Prolog, Concurrent Euclid and Maple. Other computer facilities at Oberlin include two VAX 11/750's for general use, a VAX 11/750 and a VAX 8600 for administrative work. Access to CSNET and Bitnet is anticipated to begin during the current academic year.

Oberlin is a highly selective coeducational college of arts and sciences. In comparison with other primarily undergraduate colleges, Oberlin has an extraordinarily strong record in producing students who have gone on to earn Ph.D. degrees in science and mathematics. As the world's first institution of higher education regularly to offer degrees to women and black students, Oberlin, in filling faculty positions, particularly welcomes applications from female and minority candidates.

To be assured of consideration, letters of application, including a curriculum vitae, academic transcripts, and at least three letters of reference, should be sent to George Andrews, Director, Computer Science Program, Oberlin College, Oberlin, Ohio 44074 by February 23, 1987. Application materials received after that date may be considered until the position is filled.

SUNY THE COLLEGE AT NEW PALTZ

Tenure track positions in Computer Science beginning 9/87, for faculty to support undergraduate and masters programs. Requirements: Ph.D. in Computer Science or in a related field with a Master's or equivalent in Computer Science. Evidence of excellence in teaching and scholarly achievement. College in Hudson Valley/Catskill Mountains region of New York. Departmental research includes database design, computational complexity, neural networks for AI, logic programming, operating systems, graphics. Hardware expertise desirable for one position. Send resume with names, addresses and telephone numbers of 3 references to: D. Clark, Chairman, Department of Mathematics and Computer Science, Box 10, The College at New Paltz, SUNY The College at New Paltz, New Paltz, NY 12561. Review of applications beginning 11/28/87. Pending funding approval. AA/EEO. Women and minorities urged to apply.

**SOUTHWEST TEXAS STATE UNIVERSITY
Computer Science Chairman**

The Department of Computer Science seeks a chairman to begin September 1, 1987. Located in the School of Science, the department has a faculty of 13.5 FTE with over 400 undergraduate and 80 graduate majors. Computing services are provided by two DEC 1091's, a twin VAX 8600 cluster, and a departmental MicroVax-II. Candidates must have a doctorate in computer science, or in a related field with equivalent published research. The applicant must have five years of university teaching experience, with an outstanding record of teaching, service, and leadership. The chairperson will be expected to develop strong professional relations within the university and with industrial and governmental support agencies.

The University serves an enrollment of over 19,000 in San Marcos, a city of 35,000 midway between Austin and San Antonio in the hill and lake country of Texas, and is located in a large and fast growing high technology area. Depending on qualifications, the chairperson holds rank of associate or full professor and earns \$54,492 to \$7,300 for 12 months. The closing date is January 15, 1987, and no materials postmarked after that date will be considered. The University reserves the right not to proceed with any appointment. Send letters of application, curriculum vitae, transcripts, and three current letters of reference to:

Dr. Edward Schneider, Computer Science Search Committee, School of Science, Southwest Texas State University, San Marcos, Texas 78666. SOUTHWEST TEXAS STATE UNIVERSITY IS AN EQUAL OPPORTUNITY AFFIRMATIVE ACTION EMPLOYER.

**DUKE UNIVERSITY
Department of Computer Science**

The Duke University Department of Computer Science, a 1983 recipient of an NSF CER Grant, has faculty positions available at all ranks. Applications are solicited from all areas of computer science. Applicants for senior positions must demonstrate excellence in research, while applicants for junior positions must exhibit the promise of excellence.

The Department currently has seventeen tenure track faculty, approximately 300 undergraduate majors and 50 graduate students pursuing master's and/or doctoral degrees.

The Department has major research efforts in scientific computing with emphasis on numerical linear algebra, the solution of PDEs, and VLSI simulation; computer systems with emphasis on computer architectures, modeling of fault-tolerant systems, systems performance, and communications; artificial intelligence, particularly in the areas of natural language interface, search methodologies, and expert systems; and theory and algorithms with emphasis on combinatorial and graph-theoretic studies. Special motivation for the research efforts comes from the areas of medical applications (in collaboration with the Duke Medical Center), and VLSI (in collaboration with the Microelectronics Center of North Carolina, of which Duke is a Participating Institution).

Interested applicants should send copies of their resumes and other supporting material to:

Professor Donald J. Rose
Department of Computer Science
Duke University
Durham, NC 27706

Duke University is an affirmative action, equal opportunity employer.

**UNIVERSITY OF CALIFORNIA, DAVIS
Chair, Division of Computer Science**

The College of Engineering, University of California, Davis campus, is seeking a senior faculty

member to lead its new Division of Computer Science to eminence. The Division has been allocated, initially, 15 full-time equivalent faculty positions. We are recruiting for several positions at all levels. At least two of the open positions can be filled with senior level faculty who have outstanding qualifications in the field of computer science.

Qualifications for this position include an outstanding record of research in an area of computer science, demonstrated teaching ability, sensitivity to the needs of a computer science campus, and a potential for leading the faculty to develop excellent research and instruction programs. Applicants should send a resume and letter describing their interests to:

Associate Dean Ray B. Krone
College of Engineering
University of California
Davis, CA 95616

The University of California is an Equal Opportunity/Affirmative Action employer. The position is open until filled.

**BOWDOIN COLLEGE
Chairperson, Computer Science**

The Department of Computer Science & Information Studies seeks a person to lead the development of a young discipline at a distinguished undergraduate liberal arts college. To teach two courses per semester. Commitment to excellence in undergraduate teaching and in research expected. Ph.D. in computer science preferred; applicants with Ph.D. in related field and sufficient computer science experience will be considered. Rank and tenure open.

Bowdoin College is a private, coeducational institution of 1350 students with highly selective admissions and a long tradition of excellence in scholarship. It is located in the greater Portland area along the coast of Maine in Brunswick, a community of 16,000.

College computing facilities include many microcomputers, a DEC-1091 system, and a VAX 11/780 running UNIX.

Salary and fringe benefits are competitive.

To apply, send vitae and the names of three references to Dr. Alfred H. Fuchs, Dean of the Faculty, Bowdoin College, Brunswick, Maine 04011. Applications will remain open until position is filled; review of candidates will begin January 30, 1987. Bowdoin College is committed to Equal Opportunity through Affirmative Action.

UNIVERSITY OF WASHINGTON

The University of Washington expects openings for junior and/or senior tenure-track faculty appointments in Computer Science starting in the 1987-88 academic year. A moderate teaching load allows time for both quality research and close involvement with students. As a consequence, we expect applicants to have a strong commitment to both research and teaching and an outstanding record of research for their level. A senior level applicant should have a record that would provide significant new research strength for our department. We invite applicants from all areas of Computer Science, quality being our most important selection criterion. We would particularly welcome applicants with research strength in artificial intelligence, programming languages and compilers, computer systems, and databases. In addition to the above tenure-track positions the department may have a number of visiting positions; these would require both teaching and research, and could be for any portion of the 1987-88 academic year.

Interested applicants should send a letter of application, a resume, and the names of four references to Paul Young, Chairman, Department of Computer Science FR-35, University of

Washington, Seattle, Washington 98195.

The University of Washington is an Affirmative Action/Equal Opportunity Employer. The Ph.D. is required for these positions.

**UNIVERSITY OF NORTH CAROLINA
AT WILMINGTON**

Faculty Position in Computer Science, University of North Carolina at Wilmington, Math. Sciences Department, Wilmington, NC 28403-3297. Douglas D. Smith, Chair (919 395-3291) Asst./Assoc. Professor (rank and salary commensurate with qualifications). Begins August, 1987, Ph.D. in computer science or Ph.D. in a related area with M.S. in computer science or equivalent experience. Any specialty considered. Dept. Faculty of 30 offers bachelor's degrees in computer science and in pure and applied math. VAX/VMS and UNIX are in use. Duties include teaching, scholarship and service. Deadline: Feb. 16, but applications will be received until the position is filled. An EEO/AA employer.

AURORA UNIVERSITY

Applications are invited for a faculty position in computer science with responsibilities to manage and teach within a new master degree program. Candidates should have a Ph.D. in computer science. AU is located within the high tech corridor west of Chicago. Current hardware support includes a VAX 11/780. Send resume, transcripts and three letters of reference to Homer E. Easley, Dean, School of Information Science, Aurora University, Aurora, IL 60506. AA/EEO

Moving?

**PLEASE NOTIFY
US 4 WEEKS
IN ADVANCE**

IEEE Service Center
445 Hoos Lane
Piscataway, NJ 08854

**ATTACH
LABEL
HERE**

- This notice of address change will apply to all IEEE publications to which you subscribe.
- List new address below.
- If you have a question about your subscription, place label here and clip this form to your letter.

Name (Please Print)

New Address

City

State/Country

Zip

Annual Index Volume 19, 1986



This index covers all technical items that appeared in this periodical during 1986, and items from prior years that were commented upon or corrected in 1986. The index is divided into an Author Index and a Subject Index, both arranged alphabetically.

The **Author Index** contains the primary entry for each item; this entry is listed under the name of the first author and includes coauthor names, title, location of the item, and notice of corrections and comments if any. Cross-references are given from each coauthor name to the name of the corresponding first author. The location of the item is specified by the journal name (abbreviated), year, month, and inclusive pages.

The **Subject Index** contains several entries for each item, each consisting of a subject heading, modifying phrase(s), first author's name, and enough information to locate the item. For coauthors, title, comments, and corrections if any, etc., it is necessary to refer to the primary entry in the Author Index.

AUTHOR INDEX

A

- Abdelatif, A., see Litwin, W., *C-M Dec 86 10-18*
 Adams, Glenn, see Hill, Mark, *C-M Nov 86 8-22*
 Agrawal, Dharma P., Virendra K. Janakiram, and Girish C. Pathak. Evaluating the performance of multicomputer configurations; *C-M May 86 23-37*
 Ahlberg, Raymond, see Myers, Ware, *C-M Jun 86 85-90*
 Ahn, Ilsoo, see Snodgrass, Richard, *C-M Sep 86 35-42*
 Ahuja, Sudhir, Nicholas Carriero, and David Celernter. Linda and friends; *C-M Aug 86 26-34*
 Alexandridis, Nikitas A. Adaptable software and hardware: Problems and solutions; *C-M Feb 86 29-39*

B

- Baxter, Randy D., see Stroebel, Gary J., *C-M Jul 86 42-50*
 Belady, Laszlo A. The Japanese and software: Is it a good match?; *C-M Jun 86 57-61*
 Bettinger, Mark, see Milutinović, Veljko, *C-M Oct 86 45-57*
 Bhushan, Abhay, and Michael Plass. The interpress page and document description language (Tech. commun.); *C-M Jun 86 72-77*
 Bic, Lubomir, and Jonathan P. Gilbert. Learning from AI: New trends in database technology; *C-M Mar 86 44-54*
 Blanc, Robert P. Standards—Closing the gap between OSI standards and compatible OSI products; *C-M Mar 86 97-100*

- Booth, Taylor, Tom Brubaker, J. T. Cain, Ron Danielson, Ron Hoelzeman, Glen Langdon, Dave Solder, and Murali Varamani. Design education in computer science and engineering; *C-M Jun 86 20-27*
 Bose, B. K., see Hill, Mark, *C-M Nov 86 8-22*
 Brown, David C., and B. Chandrasekaran. Knowledge and control for a mechanical design expert system; *C-M Jul 86 92-100*
 Brubaker, Tom, see Booth, Taylor, *C-M Jun 86 20-27*
 Bruno, Giorgio, Antonio Elia, and Pietro Lafase. A rule-based system to schedule production; *C-M Jul 86 32-40*
 Byington, Lester, and Doug Thies. Air Force Standard 1750A ISA is the new trend; *C-M Nov 86 50-59*

C

- Cain, J. T. Professional accreditation for the computing sciences; *C-M Jan 86 91-94*
 Cain, James T., Guest Ed., and Ronald G. Hoelzeman, Guest Ed. Introduction to special issue on computer science education in the US; *C-M Jun 86 6-9*
 Cain, J. T., see Booth, Taylor, *C-M Jun 86 20-27*
 Cardozo, Eleri, see Talukdar, Sarosh N., *C-M Jul 86 53-60*
 Carriero, Nicholas, see Ahuja, Sudhir, *C-M Aug 86 26-34*
 Carroll, John M., and Sandra A. Mazur. LisaLearning; *C-M Nov 86 35-49*
 Carter, Harold W. Computer-aided design of integrated circuits; *C-M Apr 86 19-36*
 Chandrasekaran, B., see Brown, David C., *C-M Jul 86 92-100*
 Charlesworth, Alan E., and John L. Gustafson. Introducing replicated VLSI to supercomputing: The FPS-164/MAX scientific computer; *C-M Mar 86 10-23*
 Chesley, Gilman D. From integrated circuits to integrated systems (Ltr.); *C-M Jun 86 105*
 Chien, Yi-Tsun, and Jay Liebowitz. Expert systems in the SDI environment; *C-M Jul 86 115-122*
 Christodoulakis, S., and C. Faloutsos. Design and performance considerations for an optical disk based multimedia object server; *C-M Dec 86 45-54, 56*
 Curtice, Walt, see Milutinović, Veljko, *C-M Oct 86 45-57*

D

- Dalphin, John F., see Engel, Gerald L., *C-M Jun 86 42-46*
 Danielson, Ron, see Booth, Taylor, *C-M Jun 86 20-27*
 Denney, Michael J., see Stroebel, Gary J., *C-M Jul 86 42-50*
 Dym, Clive L., see Mittal, Sanjay, *C-M Jul 86 102-114*

E

- Eggers, Susan, see Hill, Mark, *C-M Nov 86 8-22*
 Elia, Antonio, see Bruno, Giorgio, *C-M Jul 86 32-40*
 Elmami, Ramez, see Navathe, Shamkant, *C-M Jan 86 50-62*
 Engel, Gerald L., and John F. Dalphin. Vocationalism and the whole man; *C-M Jun 86 42-46*

F

- Faloutsos, C., see Christodoulakis, S., *C-M Dec 85-54, 56*
 Faught, William S., Applications of AI in engineering, *C-M Jul 86 17-27*
 Fisher, David A., and Richard M. Weatherly, Issues in the design of a distributed operating system for Ada, *C-M May 86 38-47*
 Fisher, David A., see Mundie, David A., *C-M Aug 86 20-25*
 Fox, Eric R., Kenneth J. Kiefer, Robert F. Vangen, and Shaun P. Whalen. Reduced instruction set architecture for a GaAs microprocessor system, *C-M Oct 86 71-81*
 Fura, David, see Milutinović, Veljko, *C-M Mar 86 30-42*
 Fura, David, see Milutinović, Veljko, *C-M Oct 86 45-57*

G

- Geideman, William A., see Rasset, Terrence L., *C-M Oct 86 60-68*
 Gelernter, David. Introduction to special issue on domesticating parallelism, *C-M Aug 86 12-16*
 Gelernter, David, see Ahuja, Sudhir, *C-M Aug 86 26-34*
 Gibson, Garth, see Hill, Mark, *C-M Nov 86 8-22*
 Gilbert, Barry K., Barbara A. Nausied, Daniel J. Schwab, and Rick L. Thompson. Signal processors based upon GaAs ICs: The need for a wholistic design approach, *C-M Oct 86 29-43*
 Gilbert, Jonathan P., see Bie, Lubomir, *C-M Mar 86 44-54*
 Goguen, A. Reusing and interconnecting software components, *C-M Feb 86 16-28*
 Grayson, Lawrence P. Education and America's industrial future, *C-M Jun 86 10-15, 17*
 Greiling, Paul, see Larson, Lawrence E., *C-M Oct 86 21-27*
 Gries, David, Raymond Miller, Robert Ritchie, and Paul Young. Imbalance between growth and funding in academic computer science: Two trends colliding, *C-M Sep 86 70-76*
 Gries, David. The 1984 - 85 Talbair survey, *C-M Nov 86 60-71*
 Gustafson, John L., see Charlesworth, Alan E., *C-M Mar 86 10-23*

H

- Hack, James J. Peak vs. sustained performance in highly concurrent vector machines, *C-M Sep 86 11-19*
 Halstead, Robert H., Jr. Parallel symbolic computing, *C-M Aug 86 35-43*
 Hansen, Paul, see Hill, Mark, *C-M Nov 86 8-22*
 Hayes, Philip J., see Pepper, Jeff, *C-M May 86 85-88*
 Heagerty, William, see Milutinović, Veljko, *C-M Oct 86 45-57*
 Helbig, Walter, see Milutinović, Veljko, *C-M Mar 86 30-42*
 Helbig, Walter, see Milutinović, Veljko, *C-M Oct 86 45-57*
 Hilfinger, Paul, see Hill, Mark, *C-M Nov 86 8-22*
 Hill, Mark, Susan Eggers, Jim Larus, George Taylor, Glenn Adams, B. K. Bose, Garth Gibson, Paul Hansen, Jon Keller, Shing Kong, Corinna Lee, Daecheum Lee, Jon Pendleton, Scott Ritchie, David Wood, Ben Zorn, Paul Hilfinger, Dave Hodges, Randy Katz, John Ousterhout, and Dave Patterson. Design decisions in SPUR, *C-M Nov 86 8-22*
 Hodges, Dave, see Hill, Mark, *C-M Nov 86 8-22*
 Hoelzman, Ron, see Booth, Taylor, *C-M Jun 86 20-27*
 Hoelzman, Ronald G., Guest Ed., see Cain, James T., Guest Ed., *C-M Jun 86 6-9*
 Hong, Se June, Guest Ed. Introduction to special issue on expert systems in engineering, *C-M Jul 86 12-15*
 Hoshino, Tsutomu. An invitation to the world of PA-X, *C-M May 86 68-79*
 Hudak, Paul. Para-functional programming, *C-M Aug 86 60-70*

I

- Isaak, Jim. Standards—P1003: A UNIX standard for the rest of us, *C-M May 86 80-81, 84*

J

- Jacobs, Howard. Verification of a second-generation 32-bit microprocessor, *C-M Apr 86 64-70*
 Janakiram, Virendra K., see Agrawal, Dharmia P., *C-M May 86 23-37*
 Jensen, Joseph F., see Larson, Lawrence E., *C-M Oct 86 21-27*
 Joseph, Robert L., see Pepper, Jeff, *C-M May 86 85-88*

K

- Kang, Hyunchul, see Roussopoulos, Nick, *C-M Dec 86 19-23, 25*
 Karmouch, Ahmed, see Naffah, Najah, *C-M May 86 56-66*
 Karp, Sherman, and Roosild Sven. DARPA, SDI and GaAs, *C-M Oct 86 17-19*
 Kartashev, Steven, Guest Ed., see Kartashev, Svetlana, Guest Ed., *C-M Feb 86 9-13*
 Kartashev, Steven L., see Kartashev, Svetlana P., *C-M Feb 86 61-77*
 Kartashev, Svetlana, Guest Ed., and Steven Kartashev, Guest Ed. Introduction to special issue on design for adaptability, *C-M Feb 86 9-13*
 Kartashev, Svetlana P., and Steven L. Kartashev. Adaptable software for dynamic architectures, *C-M Feb 86 61-77*
 Katz, Randy, see Hill, Mark, *C-M Nov 86 8-22*
 Keim, Kevin, see Milutinović, Veljko, *C-M Oct 86 45-57*
 Keller, Arthur M. The role of semantics in translating view updates, *C-M Jan 86 63-73*
 Keller, Jon, see Hill, Mark, *C-M Nov 86 8-22*

- Kellogg, Charles. From data management to knowledge management, *C-M Jan 86 75-84*
 Kiefer, Kenneth J., see Fox, Eric R., *C-M Oct 86 71-81*
 Knowles, Alan E., see Thakkar, Shreekanth S., *C-M May 86 8-19, 22*
 Kong, Shing, see Hill, Mark, *C-M Nov 86 8-22*
 Kuhl, Jon G., and Sudhakar M. Reddy. Fault-tolerance considerations in large, multiple-processor systems, *C-M Mar 86 56-67*
 Kuwata, Mikio, see Suzuki, Shigenobu, *C-M Apr 86 72-85*

L

- Lafaze, Pietro, see Bruno, Giorgio, *C-M Dec 86 32-40*
 Lane, John H., see Rasset, Terrence L., *C-M Oct 86 60-68*
 Langdon, Glen, see Booth, Taylor, *C-M Jun 86 20-27*
 Larson, James, see Navathe, Shamkant, *C-M Jan 86 50-62*
 Larson, James A. A visual approach to browsing in a database environment, *C-M Jan 86 62-71*
 Larson, Lawrence E., Joseph F. Jensen, and Paul T. Greiling. GaAs high-speed digital IC technology: An overview, *C-M Oct 86 21-27*
 Larus, Jim, see Hill, Mark, *C-M Nov 86 8-22*
 Leao, Luiz Alberto Villaca, see Talukdar, Sarosh N., *C-M Jul 86 53-60*
 Lee, Corinna, see Hill, Mark, *C-M Nov 86 8-22*
 Lee, Daecheum, see Hill, Mark, *C-M Nov 86 8-22*
 Liebowitz, Jay, see Chien, Yi-Tzuu, *C-M Jul 86 115-122*
 Litwin, W., and A. Abdellatif. Multidatabase interoperability, *C-M Dec 86 10-18*

M

- Maier, John H. Computer science education in the People's Republic of China, *C-M Jun 86 50-56*
 Mark, Leo, and Nick Roussopoulos. Meta-data management, *C-M Dec 86 26-36*
 Marks, Ann, see Sherman, Mark, *C-M Nov 86 32-41*
 Mazur, Sandra A., see Carroll, John M., *C-M Nov 86 35-49*
 McGregor, James, see Walker, Martin G., *C-M Apr 86 100-108*
 Mendoza-Grado, Victor, see Silbey, Alexander, *C-M Aug 86 72-85*
 Miles, Tim, see Myers, Ware, *C-M Jan 86 85-90*
 Miller, Raymond, see Gries, David, *C-M Sep 86 70-76*
 Milutinović, Veljko, David Fura, and Walter Helbig. An introduction to GaAs microprocessor architecture for VLSI, *C-M Mar 86 30-42*
 Milutinović, Veljko, see Silbey, Alexander, *C-M Aug 86 72-85*
 Milutinović, Veljko, Alexander Silbey, David Fura, Kevin Keim, Mark Bettinger, Walter Helbig, William Heagerty, Richard Zieger, Bob Schellack, and Walt Curcio. Issues of importance in designing GaAs microcomputer systems, *C-M Oct 86 45-57*
 Milutinović, Veljko, Guest Ed. Introduction to special issue on GaAs microprocessor technology, *C-M Oct 86 10-13*
 Mittal, Sanjay, Clive L. Dym, and Mahesh Morjaria. PRIDE: An expert system for the design of paper handling systems, *C-M Jul 86 102-114*
 Morjaria, Mahesh, see Mittal, Sanjay, *C-M Jul 86 102-114*
 Morshedian, Daryoush. How to fight password pirates and win (Ltr.), *C-M Jan 86 104-105*
 Mundie, David A., and David A. Fisher. Parallel processing in Ada, *C-M Aug 86 20-25*
 Myers, Brad A. A complete and efficient implementation of covered windows, *C-M Sep 86 57-67*
 Myers, Ware, Raymond Ahlberg, and Tim Miles. The competitiveness of the United States disk-drive industry, *C-M Jan 86 85-90*
 Myers, Ware. Getting the cycles out of a supercomputer, *C-M Mar 86 89-92*
 Myers, Ware. Computing sciences accreditation—Pro and con, *C-M Jun 86 47-49*
 Myers, Ware. Can software for the Strategic Defense Initiative ever be error-free?, *C-M Nov 86 61-67*

N

- Naffah, Najah, and Ahmed Karmouch. Agora—An experiment in multimedia message systems, *C-M May 86 56-66*
 Nausied, Barbara A., see Gilbert, Barry K., *C-M Oct 86 26-34*
 Navathe, Shamkant, Ramer Elmasri, and James Larson. Integrating user views in database design, *C-M Mar 86 50-62*
 Negrini, Roberto, Mariagiovanna Sami, and Renato Stefanelli. Fault tolerance techniques for array structures used in supercomputing, *C-M Feb 86 78-87*
 Newton, A. R., and A. L. Sangiovanni-Vincentelli. Computer-aided design for VLSI circuits, *C-M Apr 86 38-60*
 Niederland, Roger A., see Rasset, Terrence L., *C-M Oct 86 60-68*

O

- Ousterhout, John, see Hill, Mark, *C-M Nov 86 8-22*

P

- Padwick, Gordon, Guest Ed. Introduction to special issue on design automation (Special issue intro.), *C-M Apr 86 12-13*
 Pathak, Girish C., see Agrawal, Dharmia P., *C-M May 86 23-37*
 Patterson, Dave, see Hill, Mark, *C-M Nov 86 8-22*
 Pendleton, Joan, see Hill, Mark, *C-M Nov 86 8-22*

Pepper, Jeff, Robert L. Joseph, and Philip J. Hayes. GWW: A structured environment for building natural language interfaces (Tech. commun.); *C-M* May 85 85-88
 Pfister, Gregory F. Parallel processor project to link 512 32-bit micros (Tech. commun.); *C-M* Jan 86 98-99
 Pipitone, Frank. The FIS electronics troubleshooting system; *C-M* Jul 86 68-76
 Plass, Michael, see Bhushan, Abhay, *C-M* Jun 86 72-77
 Prywes, Noah, Yuan Shi, Boleslaw Szymanski, and Jine Tseng. Supersystem programming with Model; *C-M* Feb 86 50-60

R

Rasset, Terrence L., Roger A. Niederland, John H. Lane, and William A. Geideman. A 32-bit RISC implemented in enhancement-mode JFET GaAs; *C-M* Oct 86 60-68
 Reddy, Sudhakar M., see Kuhl, Jon G., *C-M* Mar 86 56-67
 Ritchie, Robert, see Gries, David, *C-M* Sep 86 70-76
 Ritchie, Scott, see Hill, Mark, *C-M* Nov 86 86-88
 Rosild, Sven, see Karp, Sherman, *C-M* Oct 86 17-19
 Roussopoulos, Nick, and Hyunchul Kang. Principles and techniques in the design of ADMS+; *C-M* Dec 86 19-23, 25
 Roussopoulos, Nick, see Mark, Leo, *C-M* Dec 86 26-36
 Roussopoulos, Nick, *Guest Ed.* Future directions in database systems—Architectures for information engineering (Special issue intro); *C-M* Dec 86 7-8
 Rudin, Harry. Trends in computer communications; *C-M* Nov 86 25-33

S

Sami, Mariagiovanna, see Negri, Roberto, *C-M* Feb 86 78-87
 Sangiovanni-Vincentelli, A. L., see Newton, A. R., *C-M* Apr 86 38-60
 Sanguineti, John. Performance of a message-based multiprocessor; *C-M* Sep 86 47-55
 Schellack, Bob, see Milutinović, Veljko, *C-M* Oct 86 45-57
 Schwab, Daniel J., see Gilbert, Barry K., *C-M* Oct 86 29-43
 Schwederski, Thomas, and Howard Jay, Siegel. Adaptable software for supercomputers; *C-M* Feb 86 40-47
 Shapiro, Ehud. Concurrent Prolog: A progress report; *C-M* Aug 86 44-58
 Sherman, Mark, and Ann Marks. Using low-cost workstations to investigate computer networks and distributed systems; *C-M* Jun 86 32-41
 Sherr, Sava. Standards—a new international structure for information-systems standards; *C-M* Jan 86 102-103
 Shi, Yuan, see Prywes, Noah, *C-M* Feb 86 50-60
 Shuey, Richard, and Gio Wiederhold. Data engineering and information systems; *C-M* Jan 86 18-30
 Shuey, Richard L., *Guest Ed.* Introduction to special issue on data engineering; *C-M* Jan 86 14-15
 Siegel, Howard Jay, see Schwederski, Thomas, *C-M* Feb 86 40-47
 Silbey, Alexander, Veljko Milutinović, and Victor Mendoza-Grado. A survey of advanced microprocessors and HLL computer architectures; *C-M* Aug 86 72-85
 Silbey, Alexander, see Milutinović, Veljko, *C-M* Oct 86 45-57
 Snodgrass, Richard, and Ilsoo Ahn. Temporal databases; *C-M* Sep 86 35-42
 Soldan, Dave, see Booth, Taylor, *C-M* Jun 86 20-27
 Srinl, Vison P. An architectural comparison of dataflow systems; *C-M* Mar 86 66-68
 Stefanelli, Renato, see Negri, Roberto, *C-M* Feb 86 78-87
 Stenerson, Richard O. Integrating AI into the avionics engineering environment (Tech. commun.); *C-M* Feb 86 88-91
 Stroebel, Gary J., Randy D. Baxter, and Michael J. Denney. A capacity planning expert system; IBM System/38, *C-M* Jul 86 42-50
 Su, Stanley Y. W. Modeling integrated manufacturing data with SAM*; *C-M* Jan 86 34-49
 Subrahmanyan, P. A. Synapse. An expert system for VLSI design; *C-M* Jul 86 78-89
 Sugimoto, Takao, see Suzuki, Shigenobu, *C-M* Apr 86 72-85
 Suzuki, Shigenobu, Kazutoshi Takahashi, Takao Sugimoto, and Mikio Kuwata. An integrated design system for the SX-1 and SX-2 supercomputers; *C-M* Apr 86 72-85
 Szymanski, Boleslaw, see Prywes, Noah, *C-M* Feb 86 50-60

T

Tabak, Daniel, which system is a RISC? (Ltr.); *C-M* Oct 86 85-86
 Takahashi, Kazutoshi, see Suzuki, Shigenobu, *C-M* Apr 86 72-85
 Talakdar, Sarosh N., Eleri Cardozo, and Luiz Alberto Villaca Leão. Toast: The power system operator's assistant; *C-M* Jul 86 53-60
 Tanaka, Hidehiko. A parallel inference machine; *C-M* May 86 48-54
 Taylor, George, see Hill, Mark, *C-M* Nov 86 86-88
 Thakkar, Shrekanth S., and Alan E. Knowles. A high-performance memory management scheme; *C-M* May 86 8-19, 22
 Theis, Doug, see Byington, Lester, *C-M* Nov 86 50-59
 Thompson, Rick L., see Gilbert, Barry K., *C-M* Oct 86 29-43
 Tseng, Jine, see Prywes, Noah, *C-M* Feb 86 50-60

V

Vangen, Robert F., see Fox, Eric R., *C-M* Oct 86 71-81
 Varanasi, Murali, see Booth, Taylor, *C-M* Jun 86 20-27
 Villaca Leão, Luiz Alberto, see Leão, Luiz Alberto Villaca

W

Walker, Martin G., and James McGregor. Computer-aided engineering for analog circuit design; *C-M* Apr 86 100-108
 Waxman, Ron. Hardware design languages for computer design and test; *C-M* Apr 86 90-97
 Weatherly, Richard M., see Fisher, David A., *C-M* May 86 38-47
 Whalen, Shaun P., see Fox, Eric R., *C-M* Oct 86 71-81
 Wiederhold, Gio, see Shuey, Richard, *C-M* Jan 86 18-30
 Wiederhold, Gio. Views, objects, and databases; *C-M* Dec 86 37-44
 Wood, David, see Hill, Mark, *C-M* Nov 86 86-88

Y

Young, Paul, see Gries, David, *C-M* Sep 86 70-76

Z

Zieger, Richard, see Milutinović, Veljko, *C-M* Oct 86 45-57
 Zimmermann, Philip. A proposed standard format for RSA cryptosystems; *C-M* Sep 86 21-34
 Zorn, Ben, see Hill, Mark, *C-M* Nov 86 86-88

SUBJECT INDEX

A

Access control
 how to fight password pirates and win. *Morshedian, Daryoush, C-M* Jan 86 104-105
 Accreditation
 accreditation of computer science programs; pro and con. *Myers, Ware, C-M* Jun 86 47-49
 accreditation under Computer Sciences Accreditation Boards' criteria. *Engel, Gerald L., +, C-M* Jun 86 42-46
 professional accreditation for computing sciences. *Cain, J. T., C-M* Jan 86 91-94
 Aircraft electronics
 integrating AI into avionics engineering environment. *Stenerson, Richard O., C-M* Feb 86 88-91
 Analog integrated circuits
 computer-aided engineering for analog circuit design. *Walker, Martin G., +, C-M* Apr 86 100-108
 Array processing
 fault tolerance techniques for array structures used in supercomputing. *Negri, Roberto, +, C-M* Feb 86 78-87
 peak vs. sustained performance in highly concurrent vector machines. *Hack, James J., C-M* Sep 86 11-19
 Artificial intelligence
 integrating AI into avionics engineering environment. *Stenerson, Richard O., C-M* Feb 86 88-91

C

CAD (computer-aided design); cf. Design automation
 China
 computer science education in People's Republic of China. *Maier, John H., C-M* Jun 86 50-56
 Coding/decoding; cf. Cryptography
 Computer applications; cf. Specific topic
 Computer architecture
 adaptable software for dynamic architectures. *Kartashev, Svetlana P., +, C-M* Feb 86 61-77
 survey of advanced microprocessors and HLL computer architectures. *Silbey, Alexander, +, C-M* Aug 86 72-85
 Computer architecture; cf. Specific topic
 Computer communications; cf. Computer networks
 Computer economics
 peak vs. sustained performance in highly concurrent vector machines. *Hack, James J., C-M* Sep 86 11-19
 Computer engineering education; cf. Computer science education
 Computer facilities
 capacity planning expert system for IBM System/38. *Stroebel, Gary J., +, C-M* Jul 86 42-50

Computer fault tolerance

large multiprocessor systems; fault tolerance as design issue. *Kuhl, Jon G.*, +, *C-M Mar 86 56-67*
supercomputing array structures; fault tolerance techniques. *Negrini, Roberto*, +, *C-M Feb 86 78-87*
Computer graphics; cf. Workstations
Computer graphics software
covered windows; SAPHIRE window manager. *Myers, Brad A.*, *C-M Sep 86 57-67*

Computer interfaces

GW, structured environment for building natural-language interfaces.

Pepper, Jeff, *C-M May 86 85-88*

Computer interfaces; cf. Workstations

Computer interfaces, human factors

learning Lisa; six people's problems with workstation interface. *Carroll, John M.*, +, *C-M Nov 86 35-49*

Computer languages

Ada parallel processing in its historical and technical context. *Mundie, David A.*, +, *C-M Aug 86 20-25*

case for parallel programming languages; special issue intro. *Gelernter, David*, *C-M Aug 86 12-16*

Concurrent Prolog; progress report. *Shapiro, Ehud*, *C-M Aug 86 44-58*

Interpress page and document description language. *Bhushan, Abhay*, +, *C-M Jun 86 72-77*

Linda language as parallel programming tool. *Ahuja, Sudhir*, +, *C-M Aug 86 26-34*

parafunctional approach for programming multiprocessor systems using

Parallel. *Hudak, Paul*, *C-M Aug 86 60-70*

parallel symbolic computing and Multilisp language. *Halstead, Robert H.*, *Jr.*, *C-M Aug 86 35-43*

supersystem programming with Model equational language. *Prywes, Noah*, +, *C-M Feb 86 50-60*

Computer languages; cf. Hardware design languages

Computer maintenance; cf. Computer fault tolerance

Computer networks

closing gap between OSI standards and compatible OSI products. *Blanc, Robert P.*, *C-M Mar 86 97-100*

trends in computer communications. *Rudin, Harry*, *C-M Nov 86 25-33*

undergraduate use of low-cost workstations to investigate computer

networks and distributed systems. *Sherman, Mark*, +, *C-M Jun 86 32-41*

Computer performance; cf. Multiprocessing; Supercomputers

Computer programming profession

1984 - 85 Taulbee survey on production and employment of PhDs and

faculty in computer science and engineering. *Gries, David*, *C-M Nov 86 69-71*

Computer reliability; cf. Computer fault tolerance; Software reliability

Computer science education

accreditation of computer science programs; pro and con. *Myers, Ware*, *C-M Jun 86 47-49*

accreditation under Computer Sciences Accreditation Boards' criteria.

Engel, Gerald L., +, *C-M Jun 86 42-46*

computer science education in People's Republic of China. *Maier, John H.*, *C-M Jun 86 50-56*

computer science education in the US (special issue). *C-M Jun 86 6-71*

design education in computer science and engineering. *Booth, Taylor*, +, *C-M Jun 86 20-27*

imbalance between growth and funding in academic computer science.

Gries, David, +, *C-M Sep 86 70-76*

professional accreditation for computing sciences. *Cain, J. T.*, *C-M Jun 86 91-94*

undergraduate use of low-cost workstations to investigate computer

networks and distributed systems. *Sherman, Mark*, +, *C-M Jun 86 32-41*

1984 - 85 Taulbee survey on production and employment of PhDs and

faculty in computer science and engineering. *Gries, David*, *C-M Nov 86 69-71*

Computer security; cf. Access control

Computer testing

VHDL language for computer design and test. *Waxman, Ron*, *C-M Apr 86 80-87*

Computers; cf. Distributed computing; Microcomputers; Military computers;

Parallel processing; Space-vehicle computers; Supercomputers

Cryptography

proposed standard format for RSA cryptosystems. *Zimmermann, Philip*, *C-M Sep 86 21-34*

D

Data communication; cf. Computer networks; Multiprocessing; interconnection

Data flow computing

architectural comparison of dataflow systems. *Srini, Vason P.*, *C-M Mar 86 68-88*

Data processing

hardware for spacecraft onboard processing; use of Air Force Standard

1750A. *Byington, Lester*, +, *C-M Nov 86 50-59*

Data security; cf. Computer security; Cryptography

Data structures

data engineering (special issue). *C-M Jun 86 14-84*

Databank management systems

ADMS+ database management system design; downloading database

+ Check author entry for coauthors

subsets to workstations for parallel processing. *Roussopoulos, Nick*, +, *C-M Dec 86 19-23*

interoperability of multiple databases; prototype relational multidatabase

system. *Litwin, W.*, +, *C-M Dec 86 10-18*

modeling integrated manufacturing data with SAM* (semantic association

model). *Su, Stanley Y. W.*, *C-M Jan 86 34-49*

self-describing meta-schema and update dependency formalism based on

relational model. *Mark, Leo*, +, *C-M Dec 86 26-36*

temporal support in database management systems. *Snodgrass, Richard*, +, *C-M Sep 86 35-42*

transition from data management to knowledge management. *Kellogg, Charles*, *C-M Jan 86 75-84*

Databases

AI-based trends in database technology. *Bic, Lubomir*, +, *C-M Mar 86 44-54*

future directions in database architectures for information engineering

(special issue). *C-M Dec 86 7-56*

integrating user views in database design. *Navatbe, Shamkant*, +, *C-M Jan 86 50-62*

optical-disk based multimedia object server; design and performance.

Christodoulakis, S., +, *C-M Dec 86 45-54*

role of semantics in translating view updates. *Keller, Arthur M.*, *C-M Jan 86 63-73*

Databases systems; cf. Database management systems; Information systems

Databases systems, relational

interoperability of multiple databases; prototype relational multidatabase

system. *Litwin, W.*, +, *C-M Dec 86 10-18*

object-oriented access to information stored in relational database.

Wick, G. G., *C-M Dec 86 37-44*

self-describing meta-schema and update dependency formalism based on

relational model. *Mark, Leo*, +, *C-M Dec 86 26-36*

temporal support in database management systems. *Snodgrass, Richard*, +, *C-M Sep 86 35-42*

Databases systems, searching

visual approach to browsing in database environment. *Larson, James A.*, *C-M Jun 86 62-71*

Design automation

computer-aided engineering for analog circuit design. *Walker, Martin G.*, +, *C-M Apr 86 100-108*

design automation (special issue). *C-M Apr 86 12-108*

integrated design system for SX-1 and SX-2 supercomputers. *Suzuki, Shigenobu*, +, *C-M Apr 86 72-85*

knowledge and control for mechanical design expert system. *Brown, David C.*, +, *C-M Jul 86 92-100*

PRIDE, expert system for design of paper handling systems. *Mittal, Sanjay*, +, *C-M Jul 86 102-114*

Design automation; cf. Specific topic

Design methodology

design education in computer science and engineering. *Booth, Taylor*, +, *C-M Jun 86 20-27*

Diagnosis; cf. Fault diagnosis

Digital integrated circuits; cf. Integrated circuits; MOS integrated circuits;

Very large-scale integration

Digital system education; cf. Computer science education

Digital system fault tolerance; cf. Computer fault tolerance

Digital system testing; cf. Computer testing

Disk recording; cf. Magnetic disk recording; Optical memories

Displays

covered windows; SAPHIRE window manager. *Myers, Brad A.*, *C-M Sep 86 57-67*

Distributed computing

ADMS+ database management system design; downloading database

subsets to workstations for parallel processing. *Roussopoulos, Nick*, +, *C-M Dec 86 19-23*

Agora multimedia message system for distributed, integrated electronic

office. *Naffah, Najah*, +, *C-M May 86 56-66*

DARP, distributed Ada runtime package for loosely coupled networks.

Fisher, David A., +, *C-M May 86 38-47*

Distributed systems

undergraduate use of low-cost workstations to investigate computer

networks and distributed systems. *Sherman, Mark*, +, *C-M Jun 86 32-41*

Document handling

Agora multimedia message system for distributed, integrated electronic

office. *Naffah, Najah*, +, *C-M May 86 56-66*

Interpress page and document description language. *Bhushan, Abhay*, +, *C-M Jun 86 72-77*

E

Economics; cf. RD&E economics

Education

dependence of industrial strength of US on education. *Grayson, Lawrence P.*, *C-M Jun 86 10-15*

Education; cf. Computer science education

Employment

1984 - 85 Taulbee survey on production and employment of PhDs and

faculty in computer science and engineering. *Gries, David*, *C-M Nov 86 69-71*

† Check author entry for subsequent corrections/comments

Engineering profession

1984 - 85 Taubee survey on production and employment of PhDs and faculty in computer science and engineering. *Gries, David, C-M Nov 86 69-71*

Expert systems

capacity planning expert system for IBM System/38. *Stroebel, Gary J., +, C-M Jul 86 42-50*
development and application of expert systems in engineering. *Faught, William S., C-M Jul 86 17-27*
expert systems in engineering (special issue). *C-M Jul 86 12-122*
FIS (Fault Isolation System), expert system for electronics troubleshooting. *Pipitone, Frank, C-M Jul 86 68-76*
knowledge and control for mechanical design expert system. *Brown, David C., +, C-M Jul 86 92-100*
PRIDE, expert system for design of paper handling systems. *Mittal, Sanjay, +, C-M Jul 86 102-114*
production scheduling using rule-based system. *Bruno, Giorgio, +, C-M Jul 86 32-40*
Strategic Defense Initiative; potential of expert systems. *Chien, Yi-Tzuu, +, C-M Jul 86 115-122*
Synapse, expert system for VLSI design. *Subrahmanyam, P. A., C-M Jul 86 78-89*
Toast, flexible expert system for event simulation and diagnosis in power networks. *Talukdar, Sarosh N., +, C-M Jul 86 53-60*
transition from data management to knowledge management. *Kellogg, Charles, C-M Jan 86 75-84*

F

Factory automation; cf. Manufacturing automation

Fault diagnosis

development and application of expert systems in engineering. *Faught, William S., C-M Jul 86 17-27*
FIS (Fault Isolation System), expert system for electronics troubleshooting. *Pipitone, Frank, C-M Jul 86 68-76*

Fault tolerance; cf. Computer fault tolerance

FET integrated circuits; cf. MOS integrated circuits

G

Gallium materials/devices

GaAs high-speed digital IC technology; overview. *Larson, Lawrence E., +, C-M Oct 86 21-27*
GaAs microprocessor technology; overview. *Milutinović, Veljko, Guest Ed., C-M Oct 86 10-13*
GaAs microprocessor technology (special issue). *C-M Oct 86 10-81*
microcomputer design using GaAs. *Milutinović, Veljko, +, C-M Oct 86 45-57*
military applications of GaAs technology; role of DARPA and importance for SDI. *Karp, Sherman, +, C-M Oct 86 17-19*
reduced-instruction-set GaAs microcomputer system. *Fox, Eric R., +, C-M Oct 86 71-81*
signal processors based on GaAs ICs; holistic design. *Gilbert, Barry K., +, C-M Oct 86 29-43*
32-bit RISC implemented in enhancement-mode JFET GaAs. *Rasset, Terrence L., +, C-M Oct 86 60-68*
Gallium materials/devices; cf. Signal processing

H

Hardware design languages

VHDL language for computer design and test. *Waxman, Ron, C-M Apr 86 90-97*

High-speed integrated circuits

GaAs high-speed digital IC technology; overview. *Larson, Lawrence E., +, C-M Oct 86 21-27*

Human factors; cf. Computer interfaces, human factors

I

IEEE standards

P1003 UNIX standard; overview of IEEE trial use document. *Isank, Jim, C-M May 86 80-81, 84*

Information systems

data engineering for automated information systems. *Shuey, Richard, +, C-M Jan 86 18-30*
data engineering (special issue). *C-M Jan 86 14-84*
international structure for information-systems standards. *Sherr, Sava, C-M Jan 86 102-103*
optical-disk based multimedia object server; design and performance. *Christodoulakis, S., +, C-M Dec 86 45-54, 56*

Information systems; cf. Database systems

Integrated-circuit design

computer-aided design for VLSI circuits. *Newton, A. R., +, C-M Apr 86 38-60*
computer-aided design of integrated circuits. *Carter, Harold W., C-M Apr 86 19-36*

Integrated circuits

signal processors based on GaAs ICs; holistic design. *Gilbert, Barry K., +, C-M Oct 86 29-43*
Integrated circuits; cf. Analog integrated circuits; High-speed integrated circuits; MOS integrated circuits; Very large-scale integration
Interconnection networks; cf. Multiprocessing, interconnection
International trade
competitiveness of US disk-drive industry. *Myers, Ware, +, C-M Jan 86 85-90*

J

Japan

Japanese approach to software development. *Belady, Laszio A., C-M Jan 86 57-61*
PAX (Processor Array eXperiment) supercomputer project of Japanese Ministry of Education. *Hoshino, Tautomo, C-M May 86 68-79*

K

Knowledge-based systems; cf. Expert systems

L

Languages; cf. Computer languages

Large-scale integration; cf. Very large-scale integration

Linear integrated circuits; cf. Analog integrated circuits

Logic programming

Concurrent Prolog; progress report. *Shapiro, Ehud, C-M Aug 86 44-58*
PIE, high-level-language machine for parallel logic programming. *Tanaka, Hidehiko, C-M May 86 48-54*

M

Magnetic disk recording

competitiveness of US disk-drive industry. *Myers, Ware, +, C-M Jan 86 85-90*

Maintenance; cf. Fault diagnosis

Management; cf. Production management

Manufacturing automation

development and application of expert systems in engineering. *Faught, William S., C-M Jul 86 17-27*
modeling integrated manufacturing data with SAM* (semantic association model). *Su, Stanley Y. W., C-M Jan 86 34-49*

Memories; cf. Optical memories

Memory management

high-performance memory management scheme using real-address-oriented hardware page. *Thakkar, Shreekanth S., +, C-M May 86 88-19, 22*
Microcomputer applications; cf. Specific topic

Microcomputers

design issues for GaAs microcomputer systems. *Milutinović, Veljko, +, C-M Oct 86 45-57*
reduced-instruction-set GaAs microcomputer system. *Fox, Eric R., +, C-M Oct 86 71-81*
survey of advanced microprocessors and HLL computer architectures. *Silbey, Alexander, +, C-M Aug 86 72-85*

Microprocessors

GaAs microprocessor technology; overview. *Milutinović, Veljko, Guest Ed., C-M Oct 86 10-13*
GaAs microprocessor technology (special issue). *C-M Oct 86 10-81*
GaAs VLSI microprocessor architecture. *Milutinović, Veljko, +, C-M Mar 86 30-42*
verification of second-generation 32-bit microprocessor. *Jacobs, Howard, C-M Apr 86 64-70*
32-bit RISC implemented in enhancement-mode JFET GaAs. *Rasset, Terrence L., +, C-M Oct 86 60-68*

Military computers

adaptable software and hardware; problems and solutions for military computer systems. *Alexandridis, Nikitas A., C-M Feb 86 29-39*
supersystem programming with Model equational language. *Prywes, Noah, +, C-M Feb 86 50-60*

Military standards

data-processing hardware for spacecraft; use of Air Force Standard 1750A. *Byington, Lester, +, C-M Nov 86 50-59*
Military systems; cf. Strategic Defense Initiative

MOS integrated circuits

replicated MOS VLSI components for supercomputing: FPS-164/MAX computer as example. *Charlesworth, Alan E., +, C-M Mar 86 10-23*

Multiprocessing

fault-tolerance considerations in large multiprocessor systems. *Kuhl, Jon G., +, C-M Mar 86 56-67*
performance evaluation of multiprocessor configurations. *Agrawal, Dharm P., +, C-M May 86 23-37*
performance of message-based multiprocessor. *Sanguinetti, John, C-M Sep 86 47-55*

Multiprocessing; cf. Parallel processing

Multiprocessing, interconnection

performance evaluation of multiprocessor configurations. *Agrawal, Dharm P., +, C-M May 86 23-37*

+ Check author entry for coauthors

† Check author entry for subsequent corrections/comments

N

Natural language systems

GW, structured environment for building natural-language interfaces.

Pepper, Jeff, +, C-M May 86 85-88

Networks; cf. Computer networks

Nuclear power generation

expert system applications. Faught, William S., C-M Jul 86 17-27

O

Office automation

Agora multimedia message system for distributed, integrated electronic

office. Nafah, Najah, +, C-M May 86 36-66

Optical memories

optical-disk based multimedia object server; design and performance.

Christodoulakis, S., +, C-M Dec 86 45-54, 56

P

Parallel processing

Ada parallel processing in its historical and technical context. Mundie,

David A., +, C-M Aug 86 20-25

ADMS+ database management system design; downloading database

subjects to workstations for parallel processing. Roussopoulos,

Nick, +, C-M Dec 86 19-23, 25

case for parallel programming languages; special issue intro. Gelernter,

David, C-M Aug 86 12-16

Concurrent Prolog; progress report. Shapiro, Ehud, C-M Aug 86 44-58

domesticating parallelism (special issue). C-M Aug 86 12-85

Linda language as parallel programming tool. Ahuja, Sudhir, +, C-M Aug

86 26-34

parafunctional approach for programming multiprocessor systems using

ParAlti. Hudak, Paul, C-M Aug 86 60-70

parallel symbolic computing and Multilisp language. Halstead, Robert H.,

Jr., C-M Aug 86 35-43

PIE, high-level-language machine for parallel logic programming. Tanaka,

Hideo, C-M May 86 48-54

project to link 512 32-bit microprocessors. Pfister, Gregory F., C-M Jan 86

98-99

Pipeline processing; cf. Array processing

Power system control

Tost, flexible expert system for event simulation and diagnosis in power

networks. Talukdar, Sarosh N., +, C-M Jul 86 53-60

Power generation; cf. Nuclear power generation

Privacy; cf. Cryptography

Production management

rule-based expert system for production scheduling. Bruno, Giorgio, +,

C-M Jul 86 32-40

Programming profession; cf. Computer programming profession

R

RD&E economies

imbalance between growth and funding in academic computer science.

Gries, David, +, C-M Sep 86 70-76

Reliability; cf. Software reliability

S

Scheduling; cf. Production management

SDI; cf. Strategic Defense Initiative

Security; cf. Computer security

Semiconductor devices; cf. Integrated circuits

Signal processing

GAs IC processors; holistic design. Gilbert, Barry K., +, C-M Oct 86

29-43

Signal processing; cf. Array processing

Simulation

development and application of expert systems in engineering. Faught,

William S., C-M Jul 86 17-27

Software design/development

curtailing cost of defense systems software. Kartashev, Svetlana, Guest

Ed., +, C-M Feb 86 9-13

design for adaptability (special issue). CS Feb 86 9-87

future directions in database architectures for information engineering

(special issue). C-M Dec 86 7-56

Japanese approach to software development. Belady, Laszlo A., C-M Jun 86

57-61

Software design/development; cf. Specific topic

Software development environments

GW, structured environment for building natural-language interfaces.

Pepper, Jeff, +, C-M May 86 85-88

Software education; cf. Computer science education

Software, operating systems

DARP, distributed operating system for Ada. Fisher, David A., +, C-M

May 86 38-47

P1003 UNIX standard; overview of IEEE trial use document. Isaak, Jim, C-

M May 86 80-81, 84

Software reliability

Strategic Defense Initiative; software reliability issues. Myers, Ware, C-M

Nov 86 61-67

Software reusability

adaptable software and hardware; problems and solutions for military

computer systems. Alexandridis, Nikitas A., C-M Feb 86 29-39

adaptive software for dynamic architectures. Kartashev, Svetlana P., +,

C-M Feb 86 61-77

adaptable software for supercomputers. Schwederski, Thomas, +, C-M

Feb 86 40-47

library interconnection language for reusing and interconnecting software

components. Goguen, Joseph A., C-M Feb 86 16-28

Software standards

P1003 UNIX standard; overview of IEEE trial use document. Isaak, Jim, C-

M May 86 80-81, 84

Source coding; cf. Cryptography

Space-vehicle computers

data-processing hardware for spacecraft; use of Air Force Standard 1750A.

Byington, Lester, +, C-M Nov 86 50-59

Special issues/sections

computer science education in the US. C-M Jun 86 6-71

data engineering. C-M Jan 86 14-84

design automation. C-M Apr 86 12-108

design for adaptability. CS Feb 86 9-87

domesticating parallelism. C-M Aug 86 12-85

expert systems in engineering. C-M Jul 86 12-122

future directions in database architectures for information engineering. C-M

Dec 86 7-56

GAs microprocessor technology. C-M Oct 86 10-81

Standards

closing gap between OSI standards and compatible OSI products. Blanc,

Robert P., C-M Mar 86 97-100

international structure for information-systems standards. Sherr, S., C-M

Jan 86 102-103

proposed standard format for RSA cryptosystems. Zimmermann, Philip, C-

M Sep 86 21-34

Standards; cf. IEEE standards; Military standards; Software standards

Strategic Defense Initiative

expert systems in SDI environment. Chien, Yi-Tzuu, +, C-M Jul 86

113-122

GAs technology's role in SDI. Karp, Sherman, +, C-M Oct 86 17-19

reliability of software for Strategic Defense Initiative. Myers, Ware, C-M

Nov 86 61-67

Supercomputers

adaptable software for supercomputers. Schwederski, Thomas, +, C-M

Feb 86 40-47

fault tolerance techniques for array structures used in supercomputing.

Negrini, Roberto, +, C-M Feb 86 78-87

integrated design systems for SX-1 and SX-2 supercomputers. Suzuki,

Shigenobu, +, C-M Apr 86 72-85

maximizing productivity of supercomputers. Myers, Ware, C-M Mar 86

89-92

PAX (Processor Array eXperiment) supercomputer project of Japanese

Ministry of Education. Hoshino, Tsutomu, C-M May 86 68-79

peak vs. sustained performance in highly concurrent vector machines. Hack,

James J., C-M Sep 86 11-19

replicated MOS VLSI components for supercomputing: FPS-164/MAX

computer as example. Charlesworth, Alan E., +, C-M Mar 86 10-23

U

United States

competitiveness of US disk-drive industry. Myers, Ware, +, C-M Jan 86

85-90

dependence of industrial strength of US on education. Grayson, Lawrence

P., C-M Jun 86 10-15, 17

V

Very large-scale integration

computer-aided design for VLSI circuits. Newton, A. R., +, C-M Apr 86

38-60

computer-aided design of integrated circuits. Carter, Harold W., C-M Apr

86 19-36

Synapse, expert system for VLSI design. Subrahmanyam, P. A., C-M Jul 86

78-89

verification of second-generation 32-bit microprocessor. Jacobs, Howard,

C-M Apr 86 64-70

VLSI; cf. Very large-scale integration

W

Workstations

cover windows; SAPPHIRE window manager. Myers, Brad A., C-M Sep

86 57-67

learning Lisa; six people's problems with workstation interface. Carroll,

John M., +, C-M Nov 86 35-49

SPUR (Symbolic Processing Using RISCs) multiprocessor workstation;

design. Hill, Mark, +, C-M Nov 86 8-22

undergraduate use of low-cost workstations to investigate computer

networks and distributed systems. Sherman, Mark, +, C-M Jun 86

32-41

+ Check author entry for coauthors

+ Check author entry for subsequent corrections/comments

Advertiser Index

CACI	1
Compass '87	Cover III
COMPCON '87	24
Compasac '87	5
Computer Society Membership	55
Computer Society Ombudsman	42
FTCS 17	Cover II
ICCD '87	9
Institute of Defense Analyses	17
IEEE Artificial Intelligence Applications	68-69
IEEE INFOCOM '87	57-60
IEEE Int'l Symposium on Intelligent Control	63
Index Technology Corporation	Cover IV
Interactive Software Engineering	36
Jackson Chamber of Commerce	8
LPI	43
Plenum Publishing Corp.	54
SCS	36
University of Oxford	42
USENIX	6
Classified Advertising	73-81

FOR DISPLAY ADVERTISING INFORMATION CONTACT

Southern California and Mountain States: Richard C. Faust Company, 24050 Madison Street, Suite 100, Torrance, CA 90505; (213) 373-9604.
Northern California and Pacific Northwest: Don Farris Company, 161 W. 25th Ave., #102B, San Mateo, CA 94403; (415) 349-2222.
Jack Vance, P.O. Box 3205, Saratoga, CA 95070; (408) 741-0354.
East Coast: Hart Associates, Inc., P.O. Box 339, 42 Lake Blvd., Matawan, NJ 07747; (201) 583-8500.
New England: Arpin Associates, P.O. Box 227, Weston, MA 02193; (617) 899-5613.
George Watts, III, 4 Conifer Dr., Wilbraham, MA 01095; (413) 598-4747.
Midwest: Thomas Knorr, Knorr MicroMedia, Inc., 333 North Michigan Ave., Chicago, IL 60601; (312) 726-2633.
Southwest: The House Company, 3000 Wesleyan, Suite 345, Houston, TX 77027; (713) 622-2868.

Advertising Manager: Mike Koehler, *COMPUTER Magazine*, 10662 Los Vaqueros Circle, Los Alamitos, CA 90720; (714) 821-3240, 821-8380.
 For production information, conference or classified advertising contact Carole L. Porter, *COMPUTER Magazine*, 10662 Los Vaqueros Circle, Los Alamitos, CA 90720; (714) 821-1140.

Product Index

	RS#	Page#
Apollo Computer	51	65
Borland Int'l	48	65
Compaq	40	64
Concurrent Computer	46	65
Convex Computer Corp.	47	65
Digital Electronics Systems	49	65
Eastman Communications	54	66
IBC	56	66
IBM Corp.	44	64
Index Technology Corp.	6	Cover IV
Interactive Software		
Engineering	2	36
Jackson Chamber		
of Commerce	1	8
Kinetics, Inc.	45	65
Lotus Development Corp.	43	64
LPI	4	43
McDonnell Douglas	53	66
NCR Corp.	52	66
Network Innovations	57	66
Nixdorf Computer Corp.	55	66
SCS	3	36
Sperry Corp.	41	64
Texas Instruments	50	65
Wang Laboratories	42	64

NEXT IN COMPUTER

January: Computers for AI Applications

Titles include the following: "Applications of the Connection Machine," "Initial Performance of the DADO2 Prototype," "Software Development Support for AI Programs," "The Architecture of FAIM-1," "Connectionist Architectures for Artificial Intelligence," "The Architecture of Lisp Machines," "Symbolic Architecture," "Computer Architecture for Artificial Intelligence Processing," and "What Price Smalltalk?"



For further information on advertised products, new products, or literature and to specify levels of interest in *Computer's* articles and departments fill out one of these cards. Affix stamp and mail.

Please print or type your name and address.

Reader Service Card

COMPUTER

Name _____
 Company _____
 Title _____
 Address _____
 City _____ State _____
 Zip _____ Telephone _____

Circle your level of interest in the following articles and departments:

	High	Medium	Low	Did not read
Multidatabase	101	102	103	104
Design of ADMS ±	105	106	107	108
Metadata Management	109	110	111	112
Views, Objects, and Databases	113	114	115	116
Multimedia Object Server	117	118	119	120
Open Channel	121	122	123	124
Update	125	126	127	128
New Products	129	130	131	132
New Literature	133	134	135	136
Micros	137	138	139	140
Book Reviews	141	142	143	144

Void after 6/15/87

12/86

I need more information on the following:

1	2	3	4	5	6	7	8	9	10	11	12	13	14
15	16	17	18	19	20	21	22	23	24	25	26	27	28
29	30	31	32	33	34	35	36	37	38	39	40	41	42
43	44	45	46	47	48	49	50	51	52	53	54	55	56
57	58	59	60	61	62	63	64	65	66	67	68	69	70
71	72	73	74	75	76	77	78	79	80	81	82	83	84
85	86	87	88	89	90	91	92	93	94	95	96	97	98

Products purchased or specified:

	Hobby	Job
Computers	150	151
Peripherals	152	153
Data Communication	154	155
Memories/components	156	157
Software and Services	158	159
Publications	160	161

- 170 Please send me the IEEE-CS 1986 Publications catalog.
 171 Please send me an application for advancement to senior member.
 172 Please send me an IEEE-CS membership application.

Reader Service Card

COMPUTER

Name _____
 Company _____
 Title _____
 Address _____
 City _____ State _____
 Zip _____ Telephone _____

Circle your level of interest in the following articles and departments:

	High	Medium	Low	Did not read
Multidatabase	101	102	103	104
Design of ADMS ±	105	106	107	108
Metadata Management	109	110	111	112
Views, Objects, and Databases	113	114	115	116
Multimedia Object Server	117	118	119	120
Open Channel	121	122	123	124
Update	125	126	127	128
New Products	129	130	131	132
New Literature	133	134	135	136
Micros	137	138	139	140
Book Reviews	141	142	143	144

Void after 6/15/87

12/86

I need more information on the following:

1	2	3	4	5	6	7	8	9	10	11	12	13	14
15	16	17	18	19	20	21	22	23	24	25	26	27	28
29	30	31	32	33	34	35	36	37	38	39	40	41	42
43	44	45	46	47	48	49	50	51	52	53	54	55	56
57	58	59	60	61	62	63	64	65	66	67	68	69	70
71	72	73	74	75	76	77	78	79	80	81	82	83	84
85	86	87	88	89	90	91	92	93	94	95	96	97	98

Products purchased or specified:

	Hobby	Job
Computers	150	151
Peripherals	152	153
Data Communication	154	155
Memories/components	156	157
Software and Services	158	159
Publications	160	161

- 170 Please send me the IEEE-CS 1986 Publications catalog.
 171 Please send me an application for advancement to senior member.
 172 Please send me an IEEE-CS membership application.

Comments on articles and other editorial matter

What did you find appealing about the articles/departments you liked? _____

What other topics/departments would you like to see?

General comments: _____

For reader service, see other side

This PO box for reader service cards only.

PLACE
STAMP
HERE

COMPUTER

magazine

Reader Service Inquiries
Box 24008
Los Angeles, Ca 90024
USA

Comments on articles and other editorial matter

What did you find appealing about the articles/departments you liked? _____

What other topics/departments would you like to see?

General comments: _____

For reader service, see other side

This PO box for reader service cards only.

PLACE
STAMP
HERE

COMPUTER

magazine

Reader Service Inquiries
Box 24008
Los Angeles, Ca 90024
USA

COMPASS '87 COMPUTER ASSURANCE

DATE AND LOCATION:

JULY 6-10, 1987
WASHINGTON, DC

SECOND ANNUAL CONFERENCE ON
**COMPUTER ASSURANCE: SYSTEMS INTEGRITY,
PROCESS SECURITY AND COMPUTER SYSTEM
SAFETY**

IMPORTANT DEADLINES

Three copies of abstract due:
January 31, 1987

Notification of authors:
February 28, 1987

Camera-ready manuscripts due:
April 30, 1987

CONFERENCE COMMITTEE

General Chairman:
William Wilson
NASA

Vice Chairman:
Daniel Benigni
NBS

Program Chairman:
M. Frank Houston
FDA

Publications Chairman:
John Rini
Master Systems

For more information call
Frank Houston (301)443-5020
or write:

COMPASS '87
P.O. BOX 5314
ROCKVILLE, MD 20851

Sponsored by the IEEE Washington Section with Federal military and civilian agencies and industry.

Our safety, health, and welfare as individuals and as a nation are increasingly dependent on the safe and correct use of computers. Despite advances in software engineering and system design, it is common to find major "bugs" and untrustworthy performance in critical computer-controlled systems. Existing approaches to computer assurance need to be refined technically and economically, and new approaches need to be explored.

COMPASS '86 Successfully opened a new forum for discussing computer assurance needs. The purpose of COMPASS '87 is to promote further discussion of both innovations on existing assurance methods and inventive new approaches. Discussions of software assurance in safety-critical commercial products are especially welcome. Abstracts should be three to ten pages.

Some suggested topics are:

- Safety validation
- Mathematical Methods
- Human Factors
- Applications
- Development and Management Tools
- Safety-Security Architectures
- Case Histories

We encourage authors to suggest other related topics.



Develop Better Systems Using Excelerator— Over 8,000 Systems Professionals already do.

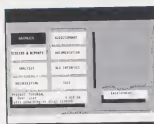
Many of the country's largest organizations have eliminated the tedious, manual process of systems specification. Instead they automate their systems analysis and design with Excelerator—the best-selling computer-aided software engineering (CASE) tool.

Their systems development teams use Excelerator in the early stages of DP projects to quickly create, analyze, and modify a design. With Excelerator's help they tackle large, critical projects such as customer billing and manufacturing requirements planning systems.

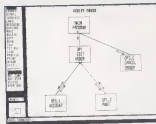
Excelerator's highly integrated, automated design environment helps them build accurate, consistent, complete, and successful specifications. Excelerator offers:

- Extensive graphics with a fully integrated dictionary
- Analysis facilities including "where used" reporting, level-to-level balancing, and extensive "ad hoc" reporting capabilities.
- Easy-to-use screen and report prototyping.
- Data sharing between analysts, projects, and host-based dictionaries.
- Documentation facilities that produce complete specifications documents.

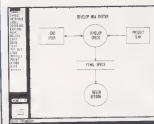
Your systems analysts and developers can join the more than 8,000 systems professionals who design better systems more efficiently—even systems



Main Menu



Structure Chart



Data Flow Diagram



Data Dictionary

Index Technology Corporation

The Leading Developer of Computer-Aided
Software Engineering products

with special requirements:

- Excelerator/RTS lets you design time- and control-critical systems. It's being used to develop systems that will fly the next generation of commercial airliners.
- For organizations with specialized development methods, we've developed Customizer which lets you tailor Excelerator and Excelerator/RTS to your own custom development environment.

Call 617-497-4473

to learn how you can develop better systems. Ask for our Excelerator information kit or order the \$30 Excelerator introductory package, which includes a 20-minute video and detailed product literature.

Excelerator®

Here's what users say:

"We cut nine months off a scheduled year-long project."

"DP employees now tend to do the kind of documentation they had not done before because it was too complex. Now documentation is done the same way every time increasing standardization."

"Communications during the analysis and design process is improved because systems analysts and end users can sit in front of the screen and review the design."